@CF

K⌠☺1M☺÷VERTEX☺CLIPPER☺L☺IBRARY☺☺0.9☺USER'S☺GUIDE☺☺PAGE☺1☺☺OF☺182P☺0☺☺☺☺☺☺☺☺☺☺☺☺☺☺

J♥╫G

TFHGABLE OF H↕G

CFHGONTENTSH↕

J♥╗G

TFHGABLE OF H↕G

CFHGONTENTS H↕G

CFHGONTINUEDH↕G

...FH

[illegible]☺☺

J↕x1MFunctions continued...Px0

J↕x1MDatabase Functions continued...Px0

J♥╗G

TFHGABLE OF H↕G

CFHGONTENTS H↕G

CFHGONTINUEDH↕G

...FH

[illegible]

J↕x1MFunctions continued...Px0

J♥╗G

TFHGABLE OF H↕G

CFHGONTENTS H↕G

CFHGONTINUEDH↕G

...FH

[illegible]

J↕x1MFunctions continued...Px0

J♥╗G

TFHGABLE OF H↕G

CFHGONTENTS H↕G

CFHGONTINUEDH↕G

...FH

≤Subject¥☺Page═H↕J☺☺

J↕x1MFunctions continued...Px0

J↕x1MVideo functions¢......................................æ☺ú☺ú☺168P

[illegible]»..................................æ☺á☺ú☺1[illegible]J↑

GENERAL INFORMATION

p1

```
IMPORTANT:Ep0☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺

p1If You want to use p0GANYH↕p1 of the VERTEX Library functions, make sure You h
p1first of all the UTINIT() function to initialize important global variables. N
p1may result in crashing Your program at any time.p0

p1If You use BLINKER have a look first of all at the sample LINK script provided
p1explains in some details what You have to look for in properly linking the lot

p1Since quite some functions are written in C or Assembler it is necessary to pu
p1CCALL object into the ROOT should You decide to OVERLAY the Library. It alsop0
p1speeds up the program.p0

▓p1C functions are marked with p0p1
```

Fp0
■p1Assembler functions are marked p0p1

```
ΣFp0
▓p1CLIPPER functions are not markedp0

p1Overview:p0☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺

p1All functions listet are written in CLIPPER 5.01 and compiled with the latest
p1compiler unless otherwise stated. Some CLIPPER functions call C- or Assemblerp
p1functions. The functions are called with or without parameters as required.p0

p1Parameters:p0☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺

p1To identify the type of parameters they are classed in different expressions o
p1the following list applies:p0

p1<expCn> an expression made up of any valid charactersp0

p1<expLn> an expression of a logical value either .T. or .F.p0

p1<expNn> an expression of a number being of type either FLOAT or INTEGERp0

p1<expAn> an expression of an array of any typep0

p1<expDn> an expression of a datep0

p1<expOn> an expression of an objectp0

p1where in all cases the lower case "n" identifies the n-th number of the releva
p1expression passed.p0

p1All parameter Idenifyers are enclosed in brackets. They may be <>, [] or () an
p1appear on their own or combined with each other. These symbols mean:p0

p1( ... )p0p1
```

```
PARAMETER LISTFp0
p1< ... >p0p1
```

MUSTFp0p1 be passedp0
p1[< ... >]p0p1

OPTIONALFp0p1î, does not need to be passedp0

J♥p1The parameters my be divided by means of a comma (,) or a vertical line (│)
p1mean:p0

p1, seperator of parameter, next to followp0

p1│ seperator of two or more possible parametersp0

p1Returnvalues:p0☺☺

p1All functions returns something unless clearly identified as returning NIL or

p1Examples:p0☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺

p1As far as I could, I included some examples with each function description. Th
p1should give You some idea as to what You could/should do with them and how to
p1them.p0

p1META☺functions:p0☺☺

p1Several times, the term META functions is being used within this Library. A ME
p1function is a complete data processing- and manipulation function made up of s
p1other functions of this Library, ready to be used and to be linked into progra
p1The use of these META functions enables one to create very quickly complexp0
p1programs without the need for long development timesp0

p1Calling☺conventions:p0☺☺

p1All functions have a 2 character prefix identifying the group into which the f
p1falls:p0

p1PrefixÖGroup╢┼☺☺☺☺p0☺☺

p1ARÖArray manipulation- and creationp0
p1BXÖBox manipulation- and creationp0
p1DBÖData manipulation- and creationp0
p1DTÖDate manipulationp0
p1FIÖFile modificationsp0
p1GRÖGraphics functionsp0
p1MFÖGeneral VERTEX META functionsp0
p1MNÖMenus and Dialog-boxes popupp0
p1MTÖMath Functionsp0
p1NXÖIndex manipulationp0
p1PRÖPrinter functionsp0
p1STÖString manipulationp0
p1SYÖlow level System functionsp0
p1UTÖGeneral Utilitiesp0
p1VIÖGeneral Video functionsp0
p1XXÖinternal VERTEX Library functionsp0

p1

IMPORTANT:Ep0p1☺☺☺☺☺☺☺☺☺☺☺Do☺not☺use☺the☺functions☺prefexed☺XX..... directly. They
âp1powerfull C- and Assemblerfunctions and should ONLY be calledp0
âp1by VERTEX functions.p0

VERTEX CLIPPER Library↕and this document are Copyright 1992 by W.A. Steinert,
All rights are reserved.

IMPORTANT COPYRIGHT INFORMATION

p1This Library VERTEX is Copyright (c) 1991-92 by the Author W.A. Steinert, as w
p1all other accompanying documents and sample programs unless explicitly statedp
p1otherwise.p0
p1Following concept applies to this package:p0

p11.You may use, copy and distribute this package freely however You may p0p1

```
NOTFp0
p1remove the Copyright Notice at the beginning of the program. You p0p1
```

```
MUSTFp0
p1mention the use of this Library in Your program and the associatedp0
p1documents.p0

x1M2.Px0p1You are encouraged to register this LIBRARY by sending a letter statin
p1wish to the author at following address:p0

p1W.A. Steinert VERTEX Enterprisesp0
p127 Dahlemstr.p0
p18750 Aschaffenburgp0
p1West Germanyp0

p13.There is a small registration fee charged to private users for this VERTEX-p
p1Library, currentlyp0
p1US $ 49.00. The user will receive a LIBRARY registered to him which will NOTp0
p1include the COPYRIGHT notice at the beginning.p0

p14.The author will inform registered users about any updates to the LIBRARYp0
p1and will give them the opportunity to update for a fee of US $ 10.00,p0
p1covering the mailing and packaging. Updates will be released about twicep0
p1annually. This policy is valid for PRIVATE USERS only. The first update isp0
p1free of charge for registered users.p0

p15.Private, registerd users may order from the author for a fee of US $ 10.00 (
p1mailing and packaging ), a registered copy of the LIBRARY in followingp0
p1languages:p0


p1FRENCHp0
p1GERMANp0
p1ENGLISHp0
p1ITALIAN ( released by 12/92 )p0

p16.Commercial users, Government offices and Companies must register thep0
p1LIBRARY prior usage however they are granted the right to use the LIBRARYp0
p1for not more than 60 days free of charge from the moment of receival of thatp0
p1LIBRARY. Any further usage of programs which have been compiled usingp0
p1this LIBRARY or the LIBRARY itself may continue only after registration ofp0
p1the VERTEX LIBRARY. The registration fee for NON-PRIVATE users is US $p0
p179.00.p0

p17.NON-PRIVATE, NON-PROFIT Organisations are encouraged to register theirp0
p1LIBRARY with the author as they will receive a registered LIBRARY free ofp0
p1charge. They may use this Library ONLY for usage associated directly withp0
p1the organisation.p0

p18.Programs compile by using a NON-REGISTERED VERTEX Library may be soldp0
p1without royalties however they MUST contain the Copyright Notice UNLESSp0
p1these programs have been compiled by using a REGISTERED COPY OF THEp0
p1VERTEX LIBRARY.p0
All rights are reserved.
VERTEX CLIPPER Library: and this document are Copyright 1992 by W.A. Steinert,
```

J♥p19.The author shall not be liable for any direct, indirect, incidental orp0
p1consequential damages resulting or seemingly result from the use, test orp0
p1distribution of the Library and/or the accompanying programs.p0

p110.The use of this Library constitutes agreement to the preceeding points asp0
p1well as to the proposition that by such You become bound to the terms ofp0
p1this agreement.p0

p1CLIPPER is a Trademark of NANTUCKET Corporation and Computer Associatesp0
=p1International Inc.p0

p1Turbo Assembler is a copyright by BORLAND INTL.p0

âp1Microsoft C is a copyright by MICROSOFT Corp.p0

```
x♥K╫AMP╢LE☺BLINKER☺LINK☺SCRIPTH¶↕☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺
p1#p0
p1# Blinker 1.50 Sample Link file for Clipper 5.01 using VERTEX Libraryp0
p1#p0
p1# This sample script is intended to be used as a template for your own scripts
p1#p0
p1# This script is set up a a development script, that enables incrementalp0
p1# linking, and overlays EXTEND.LIB from the Nantucket libraries. Whenp0
p1# incremental linking is enabled, applications will use somewhat morep0
p1# memory than normal, due to a certain amount of 'padding' being leftp0
p1# at the end of each Clipper procedure, but this is offset by thep0
p1# extremely fast incremental link times.p0
p1#p0
p1# Before shipping a completed application, you should disable thep0
p1# incremental linking feature, which will save memory, and increasep0
p1# execution speed by:p0
p1#p0
p1#a) Enabling the 5.01 Clipper paging systemp0
p1#b) Remove the padding between proceduresp0
p1#c) Turning on symbol table compressionp0
p1#p0
p1# For final links - UN-COMMENT the following command:p0
p1#p0
p1#BLINKER INCREMENTAL OFFp0
p1#DEBUGp0
p1#p0
p1# If You have troubles with Your application after linking it withp0
p1# Blinker (i.e. termiates with Memory faults ) try increasing thep0
p1# OPSIZE....p0
p1#p0
p1# Burn in your own Clipper variables here eg: F25;SWAPPATHC:\;p0
p1#p0
p1BLINKER EXECUTABLE CLIPPER F25;E1024;X64;BADCACHE;p0
p1#p0
p1# Specify the EXE filename here:p0
p1#p0
p1OUTPUT MYFILEp0
p1#p0
p1# List your Clipper S'87 object modules here. The first FILE mentioned in anyp
p1# Blinker link script MUST be a Clipper code module. The first module may bep0
p1# overlaid if required.p0
p1#p0

p1FILE MYTESTp0

p1#p0
p1# Uncomment if You want to use the profiler. Call the function UTPROFILE()p0
p1# as second function ( right after UTINIT() ) and run Your programming inp0
p1# profiling mode to find out where to put Your Modules.p0
p1#p0
p1MODULE UTINITp0
p1#MODULE UTPROFILp0
p1#p0
```

```
p1# Uncomment the next lot to increase output speed of anything which hasp0
p1# to do with GETS or Buttons.p0
p1#p0
```

```
J♥p1#MODULE UTEDPICp0
p1#p0
p1# clippers own global maxcol and maxrow function.....p0
p1MODULE __GTMAXCOLp0
p1MODULE __GTMAXROWp0
p1#p0
p1#MODULE __ATPROMPTp0
p1#MODULE BXMAKEBOXp0
p1#MODULE EXITSTATEp0
p1#MODULE READVARp0
p1#MODULE SUBSCRIPTp0
p1#MODULE _EXITSTATEp0
p1#MODULE GETACTIVEp0
p1#MODULE SETTLEp0
p1#MODULE AFILLp0
p1#MODULE BXBUTTONp0
p1#p0
p1# Uncomment and list your 3rd party non overlayable object modules herep0
p1# eg: OVERLAY's RESIDENT.OBJp0
p1#p0
p1#FILE <3rd party1>,<3rd party2>p0
p1BEGINAREAp0
p1#p0
p1#List the rest of your Clipper code files here.p0
p1#p0
p1#FILE <next Clipper file>p0
p1#  .....p0
p1#p0
p1#UNCOMMENT and list your own Clipper code libraries herep0
p1#p0
p1#p0
p1#UNCOMMENT and list your overlayable 3rd party libraries herep0
p1#as well as the VERTEX Libraryp0
p1#p0
p1#Allocates the VERTEX US-Libraryp0
p1#p0
p1  ALLOCATE VERTEXUSp0
p1# ALLOCATE <3rd party>p0
p1ENDAREAp0
p1#p0
p1# UNCOMMENT and list 3rd party non-overlayable libraries here:p0
p1# VERTEX Root Module has to be here.p0
p1#LIB <3rd party>p0
p1LIB VERTROOTp0
p1#p0
p1# If you want the 5.01 debugger UN-COMMENT the following:p0
p1# NOTE: The 5.01 debugger library MUST be specified as a FILEp0
p1#p0
p1# FILE CLD.LIBp0
p1#p0
p1# UNCOMMENT for dLESKO's FUNCky 1.5 libraryp0
p1#p0
```

```
p1#@FUNCKY50.LNKp0
p1#p0
p1# Call the distribution script to overlay EXTEND.LIBp0
p1# or substitute for LIB EXTEND.LIBp0
p1#p0
p1@CL501MIN.LNKp0
p1#p0
```

```
J♥p1# Microsoft LLIBCA C 6.00 library comes last if needed.p0
p1#p0
p1#LIB LLIBCAp0
```

IMPORTANT ARRAY FUNCTION INFORMATION

p1I have tried to put some usefull array handling functions together which might
p1handy for You.p0

p1All arrayfunctions are written entirely in CLIPPER 5.01 as most other function
p1and make full use of CLIPPERS new features inclusive codeblocks.p0

p1Whereever possible I have included some examples in the functiondescription so
p1have some idea how to use them.p0

p1Try them out, You can't break anything.p0

#♥K╓ArBigArrElem()H¶↕☺☺
KLFunction:âArBigArrElem(☺<expA>) --> integerH↕

K*Synopsis:âChecks☺an☺array☺and☺returns☺the☺length of the longest array-p0
âp1element. Useful for knowing the longest element when used inp0
âp1conjunction with arrays containing different types of elements orp0
âp1when used in conjunction with ACHOICE() to define correct displayp0
âp1if the maximum width of the box is not known.p0

K°Sample:☺âlArray☺:=☺{╓"short",;p0☺☺☺☺☺☺☺
âp1Ü╬"little longer",;p0
âp1Ü╬"very long display string" }p0
âp1llongest := ArBigArrElem(lArray)p0
âp1ltop := 0p0
âp1lleft := 0p0
âp1lbottom := 10p0
âp1lright := lleft + 1 + llongestp0
âp1lChoice := ACHOICE(ltop, lleft, lbottom, lright, lArray)p0

K←Result:☺âan☺ACHOICE☺box☺where☺all elements are visible in their entirep0
âp1length.p0

p1Notes:âWorks on arrays of all types p0p1

AND p0p1 mixed types. Trailing spaces inp0
âp1CHARACTER arrays are NOT counted, leading spaces ARE counted.p0
âp1If the array contains a codeblock, the function returns 0 asp0
âp1codeblocks are not a selectable item in themselves and should onlyp0
âp1be called via the EVAL() function.p0

p1Returns length of the longest array element or 0 if Type is other thanp0
âp1C,N,L,D or M.p0

```
Ƙ↑⊠1M☺Ö╣VERTEX☺CLIPPER☺L☺☺IBRARY☺0.9☺USER'S☺GUIDE☺☺PAGE☺15☺☺OF☺182P⊠0☺☺☺☺☺☺☺☺☺☺☺☺☺

⊠▌♥┼⊠1╓Ƙ┬⊠0J☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺
```

```
ArBldArr p1()p0
p1Function:âArBldArr( <expN1>, <expC1> ) --> array of stringsp0

p1Synopsis:âBuilds an array from a delimited stringp0

âp1Fills an array with a colon delimited string of the formp0
âp1"John:Willy:Ralph:Eddy". The colon [:] is the delimiter.p0

âp1The size of the array is set by the parameter <expN1>. If thep0
âp1number of elements defined is less than the actual amount ofp0
âp1parameters in the colon delimited string, only the amount ofp0
âp1defined elements are being created in the array and the restp0
âp1discarded.p0

âp1<expN1> Number of elementsp0

âp1<expC1> Colon delimited string of elementsp0

p1Example:âlunch := ()p0
âp1lunch := ARBLDARR( 3,"Pizzas:Chickens:Burgers" )p0

p1Returns:âArray containing the values passedp0
```

```
ARCOUNT()p1()p0☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺

p1Function:âARCOUNT( <expA>, <expC> ) --> integerp0

p1Synopsis:âCounts exact matches of TYPES in array.p0

âp1<expA>  Array to be searched.p0

âp1<expC>  value of any type to be searched for in arrayp0


p1Examples:âlFields := DBSTRUCT()p0☺☺☺☺☺☺☺☺☺☺
âp1AEVAL( lFields, { | x | AADD( lFldTypes, x[ DBS_TYPE ] ) } )p0
âp1lNbrOfChar  := ARCOUNT( lFldTypes, "C" )☺☺&& counts C fieldsp0
âp1lNbrOfNum := ARCOUNT( lFldTypes, "N" )☺☺&& counts N fieldsp0
âp1lNbrOfDate  := ARCOUNT( lFldTypes, "D" )☺☺&& counts D fieldsp0
âp1lNbrOfLogic := ARCOUNT( lFldTypes, "L" )☺☺&& counts L fieldsp0
âp1lNbrOfMemo  := ARCOUNT( lFldTypes, "M" )☺&& counts M fieldsp0
âp1lNbrOfMemo  := ARCOUNT( lFldTypes, NIL )☺&& counts NIL fieldsp0

p1Returns:ânumber of matches foundp0☺☺☺
âp10 if no matches found.p0
```

```
☺VERTEX☺CLIPPER☺L☺☺IBRARY☺0.9☺USER'S☺GUIDE☺☺PAGE☺17☺OF☺182☺☺☺☺☺☺☺☺☺☺☺☺☺
☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺
```

```
AREVENOUT()p0

p1Function:âAREVENOUT( <expA1>, <expN1> ) --> arrayp0

p1Synopsis:âExpands array elements to even length.p0

âp1<expA1> array to expandp0

âp1<expN1> width to expand each element top0

âp1This function will expand each array element to a given length. Ip0
âp1have used it to achieve an even pushbutton menu by expandingp0
âp1all elements to the biggest in the array.p0

âp1N.B. You may also shorten all elements to e specified length.p0

p1Examples:âoldarray := { "tiny",;p0
âp1Ü╬"short",;p0
âp1Ü╬"much longer" }p0

âp1newarray := AREVENOUT( oldarray, 15 )p0

âp1##------------ the new array looks like this ---------- ##p0

âp1Ü╬"tiny  "p0
âp1Ü╬"short "p0
âp1Ü╬"much longer "p0

âp1All elements are now 15 characters longp0

p1Returns:âthe new arrayp0
```

```
ARMax()

p1Function:âArMax( <expA1>, [<expN1>, <expN2>] ) --> integer│floatp0

p1Synopsis:âFinds the largest number in a NUMERICAL array.p0

âp1<expA1>  a numerical arrayp0

âp1[<expN1>,p0
âp1<expN2>] ╫optional starting and ending position for search; defaultsp0
âp1Ü╫to 1 and lastp0

p1Examples:âlNumbers := { 10, 2789, 1, 0, 9567 }p0
âp1lLargest := ARMax( lNumbers )☺☺&& lLargest = 9567p0

âp1lLargest := ARMax( lNumbers, 1, 4 )☺&& lLargest = 2789p0

p1Notes:â<expN1> must always be smaller than <expN2> and mustn't be 0.p0

p1Returns:âIf <expN1> is less than 1 or larger than <expN2>, the function willp0
âp1return NIL. Also if <expN1> is less than 1 or <expN2> is largerp0
âp1than the array. Otherwise returns the largest number.p0
```

ARMAXPOS()

p1Function: ARMAXPOS(<expA1>) --> integerp0

p1Synopsis: Finds the position of the greatest number within an array.p0

âp1<expA1> numeric array to search.p0

p1Examples: lArray := { 1, 2067, 1077, 10, 5987 }p0

âp1lPos := ARMAXPOS(lArray)&& lPos will contain 5p0

p1Returns: an integer identifying the position of the greatest number withinp0
âp1the passed array; returns 0 if an error occured.p0

```
ARMin()

p1Function:ARMin( <expA1>, [<expN1>, <expN2>] ) --> integer│float│NILp0

p1Synopsis:âFinds the smallest number in a NUMERICAL array.p0

âp1<expA1> a numerical arrayp0

âp1[<expN1>,p0
âp1<expN2>] optional starting and ending position for search; defaultsp0
âp1to 1 and lastp0

p1Examples:âlNumbers := { 10, 2789, 1, 0, 9567 }p0

âp1lSmallest := ARMin( lNumbers )☺☺&& lSmallest = 0p0

âp1lSmallest := ARMin( lNumbers, 1, 3 )☺&& lSmallest = 1p0

p1Notes:â<expN1> must always be smaller than <expN2> and mustn't be 0.p0

p1Returns:âIf <expN1> is less than 1 or larger than <expN2>, the function willp0
âp1return NIL. Also if <expN1> is less than 1 or <expN2> is largerp0
âp1than the array. Otherwise returns the smallest number.p0
```

```
ARMINPOS()¶

Function:☺ARMINPOS(☺<expA1>☺)☺☺-->☺integerp0☺

Synopsis:☺Finds☺the☺position☺of☺the☺smallest☺number within an array.p0

âp1<expA1>  numeric array to search.p0

Examples:☺lArray☺:=☺{☺2,☺2067,☺1077,☺10,☺5987 }p0

âp1lPos := ARMINPOS( lArray )&& lPos will contain 1p0

Returns:☺an☺integer☺identifying☺the☺position of the smallest number withinp0
âp1the passed array; returns 0 if an error occured.p0
```

ARSHRINKARRAY()

Function: ArShrinkArray(<expA>, <exp?>) ---> Arrayp0

Synopsis: This Function searches an array for a given expression andp0
âp1deletes this expression of the array, however leaving p0p1

NO NILFp0
âp1elements in the array where the found expression was, effectivelyp0
âp1shrinking the array by the amount of found expressions.p0

âp1<expA> a single dimension arrayp0

âp1<exp?> an expression of any kindp0

p1Note: âThe passed expression <exp?> may p0p1

NOTE☺p0p1☺be☺an☺array☺and☺thep0
âp1passed array <expA> may p0p1

ONLYFp0p1 be a single dimension arrayp0

âp1i.e.:Ütest := { 0,3,NIL,4 }p0

âp1ÜShrinkArray(test,0)p0

âp1Ü... returns an array containing { 3, NIL, 4 }p0

âp1not like the CLIPPER Function ADEL() which would leave the arrayp0
âp1at the same size with a trailing NIL element.p0

p1(Returns: an array without passed expression <exp?>p0
K(Returns☺☺☺☺☺:☺âan☺array☺without☺passed☺expres

```
ARSMLARRELEM()

p1Function:âArSmlArrElem( <expA> ) --> integerp0

p1Synopsis:âChecks an array and returns the length of the smallest array-p0
âp1element. Useful for knowing the shortest element when used inp0
âp1conjunction with arrays containing different types of elements.p0

p1Samples:âlArray := { "short",;p0
âp1Ü╫"little longer",;p0
âp1Ü╫"very long display string" }p0
âp1Ü╫lshortest := ArSmlArrElem( lArray )p0

p1Notes:âWorks on arrays of all types p0p1
```

```
ANDEp0p1☺mixed☺types.☺☺Trailing spaces inp0
âp1CHARACTER arrays are NOT counted, leading spaces ARE counted.p0
âp1If the array contains a codeblock, the function returns 0 asp0
âp1codeblock are not a selectable item in themselves and should onlyp0
âp1be called via the EVAL() function.p0

p1(Returns:☺âlength☺of☺the☺shortest☺array☺element or 0 if Type is other thanp0
âp1C,N,L,D or M .p0
```

ARSUM()

p1Function:âARSUM(<expA1>) --> floatp0

p1Synopsis:âSums all numeric elements of an array.p0

âp1<expA1> numerical array to sump0

```
p1Examples:âlArray := { 10, 11, 1, 2 }p0
âp1lTotal := ARSUM( lArray ) && lTotal contains 24p0
```

p1Returns:âa float containing the sum of the array passed, on Error NILp0

IMPORTANT BOX FUNCTION INFORMATION

p1The box functions in general make use of some very fast C and Assembler functi
p1which I have not listed at present (there is really no need for it).p0

p1You should be able to make full use of them even without knowing that there ar
p1C or Assembler functions behind all this. These C and Assembler functions are
p1behaved and can all be overlayed (at least I had no problems sofar).p0

p1All box functions make use wherever possible of all TEXT modes of Your video c
p1However I am unable to test the functions on all cards available, so if there
p1problem with a specific card, let me know what card You have and what the prob
p1is. I try to include the fixup in the next releases. Remember, licensing the L
p1will guarantee You a bug-fix before the next release, free of charge!p0

p1

IMPORTANT:Ep0☺☺

p1All BX.... functions make heavy use of ROM BIOS function calls and use the IBM
p1character set to draw the boxes. If You decide to switch Your Graphics card in
p1GRAPHICS MODE remember that all characters will be display exclusive OR'd (XOR
p1with the current display contents. They make no use of the attributes You may
p1Thats why the frame around boxes displayed in Graphics mode on an empty blackp
p1screen will have ALLWAYS black as background color.p0

p1I have been using BLINKER 1.51 for putting together the sample programs and Ip
p1think RTLINK or WARPLINK should be able to handle them just as well.p0

```
■♥K╫BXBOX()J♣¶↕☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺

K1Function:âBXBOX(╪<expN1>..<expN4>, [<expN5>], [<expN6>],[<expN7>],p0
âp1Ü╪[<expN8>], [<expN9>] ) --> logicalp0

K1Synopsis:âPops☺a☺Box☺on☺given☺coordinatesp0☺☺

âp1<expN1>..<expN4> Box coordinates on screenp0

âp1[<expN5>] Boxtype as identified below...p0

âp1[<expN6>] attribute of Boxp0

âp1[<expN7>] shadow positionp0

âp1[<expN8>] shadow foreground colorp0

âp1[<expN9>] shadow background colorp0

âp1[<expC1>] user defined character for boxframep0

âp1Boxtype definition for <expN5> being -1 to 15:p0
dßâGBoxÜType╪Bottom╔RightßTop►☺LeftH↕☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺
âp1-1Ü(----)╔-ß·-►☺☺-☺☺-p0
âp10Ü(0000)╔Sß·S►☺☺S☺☺Sp0
âp11Ü(0001)╔Sß·S►☺☺S☺☺Dp0
âp12Ü(0010)╔Sß·S►☺☺D☺☺Sp0
âp13Ü(0011)╔Sß·S►☺☺D☺☺Dp0
âp14Ü(0100)╔Sß·D►☺☺S☺☺Sp0
âp15Ü(0101)╔Sß·D►☺☺S☺☺Dp0
âp16Ü(0110)╔Sß·D►☺☺D☺☺Sp0
âp17Ü(0111)╔Sß·D►☺☺D☺☺Dp0
âp18Ü(1000)╔Dß·S►☺☺S☺☺Sp0
âp19Ü(1001)╔Dß·S►☺☺S☺☺Dp0
âp110Ü(1010)╔Dß·S►☺☺D☺☺Sp0
âp111Ü(1011)╔Dß·S►☺☺D☺☺Dp0
âp112Ü(1100)╔Dß·D►☺☺S☺☺Sp0
âp113Ü(1101)╔Dß·D►☺☺S☺☺Dp0
âp114Ü(1110)╔Dß·D►☺☺D☺☺Sp0
âp115Ü(1111)╔Dß·D►☺☺D☺☺Dp0

âp1If a box type is chosen from this table, corner characters will bep0
âp1automatically chosen to match the sides.p0
âp1-1 will allow user defineable box character to be used.p0

K1Notes:âWorks☺best☺in☺TEXT☺modes only as You are unable to usep0
âp1attributes to display characters. As these box drawing  functionsp0
âp1are done with characters, they will use colors only.p0

K1Returns:☺â.T.☺if☺successfully☺created,☺otherwise .F.p0
```

```
╪☻╫K╪BXPUSHBUTTON()H¶↕☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺

p1Function:âBxPushButton( <expN1> .. <expN3>, <expA1>, [<expL1>] ) -->p0
âp1integerp0

p1Synopsis:âSets up a number of HORIZONTAL or VERTICAL pushbuttons andp0
âp1waits for user input.p0

âp1<expN1> .. <expN2> row/column coordinates of pushbottons.p0

âp1<expN3>  button number to be used to start off with.p0

âp1<expA1>  array of strings defining the buttons.p0

âp1[<expL1>]╪defines if buttons are displayed horizontal (.T.) orp0
âp1vertical (.F.);p0
âp1Ü╪defaults to .T.p0

âp1The function selects the first character of each string defined inp0
âp1<expA1> as a shortcut key so selection may be done either byp0
âp1using the first character of each string or the cursor controlp0
âp1keys and ENTER. Colors used are in accord with the currentp0
âp1SETCOLOR() settings.p0

âp1This function is a low level function and used by all boxes whichp0
âp1require user input. I have left it out for use in other ways i.e. sop0
âp1you can create your own boxes with user response.p0


K1Examples:âlStart☺:=☺1p0☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺
âp1lChoices := { "YES","MAYBE","NO" }p0
âp1lSelected := bxhpushbutton( 10, 20, lStart, lChoices )p0

âp1DO CASEp0
âp1ÜCASE lSelected == 1►☺&& yesp0
âp1Ü╪.... your code ...p0
âp1ÜCASE lSelected == 2►☺&& maybep0
âp1Ü╪....p0
âp1ÜCASE lSelected == 3►☺&& nop0
âp1Ü╪....p0
âp1ENDCASEp0

K1(Returns:âan☺integer☺identifying☺the☺selected choice, otherwise 0p0
```

BXMAKEBOX()

Function: BXMAKEBOX(<expN1>..<expN4>, [expC], [expN5]) --> stringp0

Synopsis: Draws a shadow box on the screen at given coordinates and savesp0
âp1the underlying screen area as well as the current colorsettings.p0

âp1<expN1>..<exp4> Screen coordinates of the box to be drawn.p0

âp1The coordinates p0p1

```
DO NOTFp0p1 include the drop shadow. These arep0
âp1automatically included, so make sure you have enough space forp0
âp1them. The shadow is 1 extra character position depending uponp0
âp1the placing of the shadow.p0

âp1<expC>   an optional colorstring or color variable for the popup ▐☺p0
âp1Ü╫box.p0

âp1<expN5>  an optional number defining the shadow position. You └☺▐☺p0
âp1Ü╫can determine thep0
âp1Ü╫Shadow position on hand of the numeric keypad on the ▐☺p0
âp1Ü╫Keyboard:p0

âp1Ü╫7  =  top leftp0
âp1Ü╫9  =  top rightp0
âp1Ü╫1  =  bottom leftp0
âp1Ü╫3  =  bottom rightp0
âp1Ü╫0  =  no shadowp0

p1Examples:âlScrSave := BXMAKEBOX(5,5,10,10)p0
âp1Ü* draws a box at 5,5,10,10. Color is as defined by SETCOLOR()p0
âp1Ü* shadow position is based on C_SHADPOS.p0

âp1lScrSave := BXMAKEBOX(5,5,10,10,C_POPCOL)p0
âp1Ü* draws a box at 5,5,10,10. Color is C_POPCOL.p0
âp1Ü* shadow position is based on C_SHADPOS.p0

âp1lScrSave := BXMAKEBOX(5,5,10,10,C_POPCOL,9)p0
âp1Ü* draws a box at 5,5,10,10. Color is as defined by SETCOLOR()p0
âp1Ü* shadow position is 9 (upper right hand corner)p0

p1Notes:âp0p1
```

Do not use the CLIPPER Function RESTSCREEN() to restore savedFp0
âp1

screenareas which are stored in the string returned by theFp0
âp1

BXMAKEBOX() function.Fp0p1 Use the provided BXUNBOX() functionp0
âp1instead. If You use this function in Graphics mode, remember thatp0
âp1You can't use attributes to display characters. In Graphics modep0
âp1characters are displayed with COLOR, not ATTRIBUTES.p0

p1Returns a string containing the coordinates, colorsettings and underlyingp0
âp1screenarea.p0

BXSCROLL()

Function: BXSCROLL(<expN1> , <expN7>) --> logicalp0

Synopsis: Scroll area of given box coordinatesp0

âp1<expN1> number of lines to scroll (0 clears the area within thep0
âp1parametersp0
âp1Ü<expN3> to <expN6>.p0

âp1<expN2> attribute to use for scrollp0

âp1<expN3> .. <expN6>ßarea coordinates to scroll (top, left, bottom,p0
âp1right)p0

âp1<expN7> direction to scroll (1 = up, 0 = down)p0

âp1This function will scroll a given area within a box either up orp0
âp1down using a given attribute. It scrolls the area a number ofp0
âp1lines.p0

Returns: .T. if successful, otherwise .F.p0

```
BXUNBOX()

K1Function:âBXUNBOX( <expC> )p0

K1Synopsis:âRemoves a box which was previously created by the functionp0
âp1BXMAKEBOX()p0

âp1<expC>  Saved screen string created by BXMAKEBOX()p0

K1Examples:âlScrSave := BXMAKEBOX( top, left, bottom, right )p0

âp1... your code ...ß&& lScrSave contains the saved screenp0

âp1.................p0

âp1BXUNBOX( lScrSave )p0

p1Notes:âp0p1
```

Do not use with any other saved screen string than the oneFp0
âp1

created by the BXMAKEBOX() function.Fp0

p1(Returns:âNothingp0☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺

IMPORTANT DATABASE FUNCTION INFORMATION

p1There are a lot of extra DB functions NANTUCKET did not supply so I have tried
p1close the gap a little.p0

p1Some of the DB Functions should really be classed as META functions (where do
p1draw the line?), but I left them here.p0

p1All DB Functions make extensive use of codeblocks so they should be rather fas
p1Whereeever I could I also made use of lots of CLIPPERS Objects, especially thep
p1TBROWSE objects as You may find in the DBTBLVIEW() function.p0

DBBIGFIELD()

p1Function:âDBBigField(<expC>) --> integerp0

p1Synopsis:âChecks a database and finds the length of the longest field withinp0
âp1that database.p0

p1â<expC> the ALIAS or the NAME of the database to searchp0

p1âThis is a very useful funtion if you don't know the length of thep0
âp1fields within a database and you want to display the contents of ap0
âp1database on the screen. Use this function to find out how longp0
âp1the longest Field is.p0

p1âChecks if the database is in use and searches it. Opens thep0
âp1database if not in use in the next lower unoccupied workarea andp0
âp1searches it.p0

p1Examples:âlDbase := "MYDATA"p0
p1âlongfield := DbBigField(lDbase)p0

p1Result:â"longfield" contains a numeric value specifying the length of thep0
âp1longest field.p0

p1Returns:âan integer specifying the length of the longest field in thep0
âp1database p0p1

(this is NOT the length of the fieldname)Fp0

```
K1 1M  VERTEX CLIPPER LIBRARY 0.9 USER'S GUIDE  PAGE 33 OF 182 x0

KEYWORD ( )

K1 Function: DbConWord( <expC>, <expL> ) --> logicalp0

K1 Synopsis: Function converts a CLIPPER or DBase File into a MS-WORDp0
âp1Datafile, that is, it actually copies the Data into a newly createdp0
âp1MS-WORD Datafile.p0

p1âRequires a database to be in use in the current workarea.p0

p1â<expC>   contains the Filename of the MS-WORD Datafile to bep0
âp1createt.p0

p1â<expL>   logical expression denotes if erased records are to bep0
âp1copied (.F.)  or not (.T.). Defaults to (.T.) ... that is - erasedp0
âp1records are NOT copied.p0

K1 Returns: .T. if successfully transfered, .F. on error.p0
```

DBDELREC()

p1Function:âDBDELREC([<expC>], [<expN>]) --> integerp0

p1Synopsis:âPops up a dialog box to allow deletion or recalling of the currentp0
âp1record.p0

p1âThe prompt in the box depends upon the current status of thep0
âp1record. If the record is currently deleted, the prompts will be:p0

p1âÜUn-Deletep0
âp1ÜNo Actionp0

p1âotherwisep0

p1âÜDeletep0
âp1ÜNo Actionp0

p1â[<expC>] optional box colorstring; defaults to C_POPCOLp0

p1â[<expN>] optional shadowposition; defaults to C_SHADPOSp0

p1Examples:â1WhatHappened := DBDELREC()p0

p1Returns:â1 if record was deletedp0
p1â-1 if record was restoredp0
p1â0 if nothing happenedp0

```
DBEXPDATA()

Function: DbExpData( [<expC>], [<expN1>], [<expN2>, <expN3>] ) --> NILp0

Synopsis: This function gives the user a choice of exporting a set of datap0
âp1previously defined. The function works on the current workareap0
âp1and exports ALL UNDELETED records found.p0

p1âTo export only part of the data, create a temporary database andp0
âp1use the function on it.p0

p1â[<expC>] optional box color string; defaults to C_POPCOLp0

p1â[<expN1>]optional shadow; defaults to C_SHADPOSp0

p1â[<expN2>..<expN3>]ßoptional top/left coordinates of box; defaultsp0
âp1center.p0

p1âCurrently supported export formats are:p0

╙p1DBASE III plusp0
εp1SDFp0
╙p1ASCII delimitedp0
╬p1MS-WORD Datafilep0

p1âothers are in preperation.p0

p1âDuring the operation a box is being presented showing thep0
âp1progress by displaying the amount of records still to bep0
âp1processed.p0

Note: a workarea and database must be in use before this function isp0
âp1called.p0

Examples: USE EXPDBp0
p1âDBEXPDATA()p0

p1â- or -p0

p1âDBEXPDATA( Color, Shadow, top, Left )p0

Returns: Nothingp0
```

DBFLDEXIST()

```
p1Function: DBFLDEXIST( <expC1>, <expC2> ) ---> logicalp0

p1Synopsis: Checks if Field exist in Database.p0

p1âThe function will check if a given fieldname exist in a sepcifiedp0
âp1database.p0

p1â<expC1> database to search for fieldp0

p1â<expC2> fieldname to search forp0

p1Examples: ? DBFLDEXIST( "MYDBASE.DBF", "NAMES" )p0

p1âÜ&& prints .T. if "NAMES" exist in "MYDBASE.DBF"p0

p1Returns: .T. if field exists,p0
p1âotherwise .F.p0
```

DBIMPDATA()

```
K1Function:âDbImpData( [<expC>], [<expN1>], [<expN2>, <expN3>] ) --> NILp0

K1Synopsis:âThis function gives the user a choice of importing a set of data.p0
âp1The function works on the current workarea and imports ALL datap0
âp1of the given file.p0

p1â[<expC>] optional box color string; defaults to C_POPCOLp0

p1â[<expN1>]optional shadow; defaults to C_SHADPOSp0

p1â[<expN2>..<expN3>]ßoptional top/left coordinates of box; defaultsp0
âp1center.p0

p1âCurrently supported import formats are:p0

╙p1DBASE III plusp0
εp1SDFp0
╙p1ASCII delimitedp0

p1âDuring the operation a box is being presented showing thep0
âp1progress by displaying the amount of records imported.p0

K1Note:âa workarea and database must be in use before this function isp0
âp1called.p0

K1Examples:âUSE EXPDBp0
p1âDBIMPDATA()p0

p1â- or -p0

p1âDBIMPDATA( Color, Shadow, top, Left )p0

K1Returns:âNothingp0
```

```
K1☺1MÖ╢VERTEX☺CLIPPER☺L☺☺IBRARY☺0.9☺USER'S☺GUIDE☺☺PAGE☺38☺☺OF☺182p☺0☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺

#♥K╢p╢DBINUSE()p¶☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺

K1Function:âDbInUse(☺<expC1>,☺[<expN1>]☺)☺--->☺Workareap0

K1Synopsis:âSearches☺undestructive☺in☺all☺Workareas if the ALIAS passed inp0
âp1<expC> is already in Use.p0

p1â<expC1> Alias to be searched for.p0

p1â[<expN1>] optional starting workarea.p0

K1Notes:âThe☺Function☺will☺terminate as soon as the first ALIAS was found.p0

K1Returns:âWorkarea☺if☺ALIAS☺was☺found,☺0☺if ALIAS was not found.p0
```

[illegible]

```
p1Function:âDBISDBF( <expC1> ) --> logicalp0

p1Synopsis:âChecks if Database is valid DBASE format.p0

p1âThe function tests the database in question by trying to open it.p0
âp1On success, the database will be closed and the old settingsp0
âp1reinstated - returning .T.; on error the function will reset the oldp0
âp1settings and return .F.;p0

p1âThat the function doesn't crash is made possible by means of thep0
âp1VERTEX Error handling function.p0

p1Examples:âlisvalid := dbisdbf( "SAMPLE.DBF" )p0
p1  âÜ&& lisvalid contains .T. if valid databasep0

p1Returns:  â.T. if database is valid and not corruptedp0
p1âotherwise .F.p0
```

DBISMEMO()

p1Function:âDBISMEMO() --> logicalp0

p1Synopsis:âChecks if there exists a MEMO field for a database.p0

p1Examples:âDBISMEMO()p0

p1â - or -p0

p1âALIAS->(DBISMEMO())p0

p1Returns:âT. if there exists a memo field for the database;p0
p1âotherwise .F.p0

```
K1M VERTEX CLIPPER LIBRARY 0.9 USER'S GUIDE  PAGE 41 OF 182p0

DBJUMP()

p1Function:âDBJUMP() ---> stringp0

p1Synopsis:âPops up available, open Databases for selection.p0

p1âThe function searches all possible workareas for open databasesp0
âp1and pops up a selection of available databases. The user mayp0
âp1select a database which will become then the current database.p0

p1Examples:âUSE customerp0
p1âUSE addressesp0
p1âDBJUMP()&& pops a selection ofp0
p1âÜ&& CUSTOMER and ADDRESSESp0
p1âÜ&& allowing the user to changep0
p1âÜ&& the workarea to any of themp0

p1Returns:âthe alias of the selected workarea if changed successful, otherwisep0
âp1the alias of the current workarea.p0
```

[illegible]

Function: DbLookFor(<expC1>) ---> Alias

Synopsis: Searches a database for records fitting a search-expression.p0

p1â<expC1> Search expression which must evaluate to a logical valuep0

p1âIt searches the Database of the current Workarea and will writep0
âp1the Recordnumber of the found Records in a temporary File.p0

Filename: TMP#####.### where # is a numeric valuep0

Structure: Fieldname --> FOUNDRECp0
p1âFieldtype -> NUMERICp0
p1âFieldlength ->10p0

ALIAS: RECFOUNDp0

p1âRecordnumbers are written in logical order of any found records.p0
âp1If any records have been found the Alias of the temporaryp0
âp1Database will be passed, otherwise an empty String is passed.p0

Notes: Apart from type-checking of the passed parameter and thep0
âp1amount of parameters passed, no other check is done. Anp0
âp1incorrect expression results in an error. Best use the functionp0
âp1MFMAKEEXPR() to create interactively a VALID expression.p0

p1âThe Database for the found records is opened in EXCLUSIVE modep0
âp1in the next available lowest workarea and remains opened. If ap0
âp1call to this function has been done previously, the same workareap0
âp1will be used, all previously found records dropped and a new setp0
âp1created.p0

p1âProcesses the database in its natural order unless Indexes are inp0
âp1use. In this case the Index-order is used.p0

NETWORK USERS: Be aware that this functions does not lock the found records in
âp1any way. This means that other users may delete the foundp0
âp1records after the function terminates and you are not aware ofp0
âp1this - thus it may lead to an erroneous result.p0

Returns: Alias of database where found recordnumbers are listed forp0
âp1further processingp0

DBSELDBVAL()

```
Function: DbSelDbVal( <expC1>, <expN1>..<expN4>, [<expC2>], [<expC3>] ) --p0
âp1> Fieldvaluep0

Synopsis: A function which returns a fieldvalue from a database chosen byp0
âp1a user.p0

p1â<expC1>  a valid fieldname of a database in use in the current p0
âp1workarea.p0

p1â<expN1>..<expN4>ßvalid coordinates of the window in which a p0
âp1ßselection ofp0
âp1ßfieldvalues appears.p0

p1â[<expC2>] an optional parameter used as header of the window.p0

p1â[<expC3>] optional displaycolor stringp0

p1â[<expN4>] optional shadow positionp0

p1âThe window will be created and a selection of fieldvalues willp0
âp1appear. The order of the values depends upon the index currentlyp0
âp1in use or natural order if no index is in use. A highlight bar isp0
âp1used to scan the values by means of the Cursor control keys asp0
âp1well as PgUP and PgDN. This function requires the used databasep0
âp1to be opened in shared modus.p0

Notes: The fieldwidth will be adjusted to the size of the window. Itp0
âp1means that fieldvalues larger than the window will be truncated top0
âp1fit the window. This is of particular interest for NUMERIC fieldsp0
âp1since the number displayed may not be the correct number: e.g.p0

[illegible]

p1âalthough only part of the fieldvalue is being displayed, thep0
âp1complete fieldvalue will be returned. Make sure that the windowp0
âp1for numeric fields wide enough to allow the user to select thep0
âp1right number.p0

p1âThe fieldname passed must be a valid fieldname for the databasep0
âp1in use. No parameter checking is done apart from correct type.p0

Return: fieldvalue if successful empty string if notp0
```

```
K1☺1Mö╢VERTEX☺CLIPPER☺L☺☺IBRARY☺0☺9☺USER'S☺GUIDE☺☺PAGE☺44☺OF☺182Px0☺☺☺☺☺☺☺☺☺☺☺☺☺☺


DBSHRINKFILE()H¶↕☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺

K1Function☺âDBShrinkFile()☺--->☺logicalp0☺☺☺☺☺☺

K1Synopsis☺âPacks☺a☺Database☺and☺updates☺associated indexes.p0

p1âRequires an open database in the current workarea. All openp0
âp1Indexes will be updated during the packing process:p0


K1Notes☺☺âDatabase☺will☺be☺locked during packing process.p0

K1Returns☺â.T.☺if☺packing☺process☺was☺successful, .F. if aborted orp0
âp1unsuccessful of whatever reason. These reasons could be:p0

p1â1)  Database not opened before functioncallp0
p1â2)  Database could not be locked and user terminates function.p0
```

DBTBLVIEW()

```
Function: DBTBLVIEW( <expC1>, [<expC2>, <expN1>..<expN4>],[<expL1>] )
[illegible] TBROWSEP
<expC1>  a databasename to work on ( this is NOT the ALIAS )
<expC2>  an optional index to be active while working with the
<expN1>..<expN4> optional, [illegible] MAXROW()-3, MAXCOL()-3 [illegible] the Table;
[<expL1>] optional logical; default .F. [illegible] ing, .T. edits,
Returns: .T. Data was worked on, all ok., .F. Parameter error
```

DBFLDSUM()

p1Function:âDBFLDSUM(<expC1> | <expN1>) --> floatp0

p1Synopsis:âSum a Field of a database currently in use.p0

p1âThe function is adding up all values of a database field eitherp0
âp1passed by NAME or FIELDPOSITION.p0

p1â<expC1> name of databasefield to sump0

p1â<expN1> fieldposition of databasefield to sump0

p1Examples:âUSE ACCOUNTSp0
p1âlTotal := DBFLDSUM(2)p0

p1â - or -p0

p1âlTotal := DBFLDSUM("PAYMENT")p0

p1âboth do the same and return the same value.p0

p1Returns:âa float number representing the TOTAL of the Field,p0
p1â0 on error.p0

```
DBZAP()

p1Function:âDbZap() --> logicalp0

p1Synopsis:âZaps a database, indexes and all memofields currently in usep0

p1âThe Function takes care of Network locking itself.p0

p1Examples:âUSE CUSTOMER INDEX SNAMEp0
p1âlResult := DBZAP()p0
p1âÜ╪  && lResult contains .T. if zapping wasp0
p1âÜ╪  && successful. Index SNAME is also zappedp0

p1Returns:â.T. if successfulp0
p1â.F. if unsuccessful for whatever reasonp0
```

IMPORTANT DATE FUNCTION INFORMATION

p1CLIPPER somehow neglected the DATE functionality and if You need some DATEp0
p1Functions.... well here they are.p0

p1For the Date calculations, all DATE Functions will temporarily change the DATE
p1ANSI standard, do the calculations and then switch back to whatever DATEFORMAT
p1You choose to run Your system on.p0

p1I found this the most reliable way of calculating DATES so one knows where the
p1and the MONTHS etc. are...p0

```
DTDATEADD()

p,Function:âDTDATEADD( <expD>, <expN1>, <expN2> ) --> Datep0

p1Synopsis:âAdds or subtracts days, weeks, months or years to/from a givenp0
âp1date.p0

p1â<expD>  date to work onp0

p1â<expN1>  a positive or negative integer to add or subtract fromp0
âp1date <expD>p0

p1â<expN2> action as defined in VERTEX.CH:p0

x1MâÜPx0p1DT_DAY= add daysp0
x1MâÜPx0p1DT_WEEK  = add weeksp0
x1MâÜPx0p1DT_MONTH = add monthsp0
x1MâÜPx0p1DT_YEAR  = add yearsp0

p1Examples:âDTDATEADD( DATE(), 5, DT_DAY )☺&& adds 5 daysp0
p1âDTDATEADD( DATE(), -5, DT_DAY )☺&& subtract 5 daysp0
p1âDTDATEADD( DATE(), 5, DT_MONTH )☺&& DTs 5 monthsp0
p1âDTDATEADD( DATE(), -5, DT_MONTH )☺&& subtract 5 monthsp0

p1Notes:âIf the number <expN1> is of a positive value, it will be added top0
âp1the date in <expD>; if the number <expN1> is of a negative value,p0
âp1it will be subtracted. The function takes care itself of leapyears.p0

p(Returns:âa date as result of the operation,p0
p1âNIL if wrong parameters were passedp0
```

```
K1x1MÖ╫VERTEX☺CLIPPER☺L☺☺IBRARY☺0☺.9☺USER'S☺GUIDE☺☺PAGE☺50☺☺OF☺182Px0☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺


DTBEGEND()¶☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺

K1Function:âDTBEGEND(☺<expD>,☺<expN1>,☺<expN2>, [<expN3>] ) --> Datep0

K1Synopsis:âDetermines☺begin☺or☺end☺of☺week,☺month or quarterp0

p1â<expD>  Date to examinep0

p1â<expN1>  Beginning ( WEEK_BEGIN ) orp0
p1âÜ╬End ( WEEK_END ) of ...p0

p1â<expN2>  Week ( OF_WEEK ), or Month ( OF_MONTH ),p0
âp1Ü╬or Quarter ( OF_QUARTER ) in which...p0
p1âÜ╬input Date <expD> falls.p0

x1MâPx0p1WEEK_BEGIN, WEEK_END, OF_WEEK, OF_MONTH, OF_QUARTER arep0
âp1Macros defined in VERTEX.CHp0

K1Options:âIf☺week,☺<expN3>☺is☺day☺of☺the week beginning or end of the weekp0
âp1<expN1> falls on, with sunday being 1 and saturday being 7.p0

K1Examples:âdate☺:=☺ctod("01/15/90")p0☺☺☺☺☺☺☺☺☺☺
p1âbow:= DTBEGEND( cdate, WEEK_BEGIN, OF_WEEK, 1 )p0
p1âÜ╬&& beg of weekp0

p1âeow:= DTBEGEND( cdate, WEEK_END, OF_WEEK, 7 )p0
p1âÜ╬&& end of weekp0

p1âbom:= DTBEGEND( cdate, WEEK_BEGIN, OF_MONTH )p0
p1âÜ╬&& beg of monthp0

p1âeow:= DTBEGEND( cdate, WEEK_END, OF_MONTH )p0
p1âÜ╬&& end of monthp0

p1âboq:= DTBEGEND( cdate, WEEK_BEGIN, OF_QUARTER )p0
p1âÜ╬&& beg of quarterp0

p1âeoq:= DTBEGEND( cdate, WEEK_END, OF_QUARTER )p0
p1âÜ╬&& end of quarterp0

K1Returns:☺âa☺date☺depending☺upon☺the☺action donep0
```

```
■♦x1╟DTBOM()♦¶☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺

p,Function:âDTBOM(☺[<expD>]☺)☺--☺>☺datep0☺☺☺☺☺☺☺☺

p1Synopsis:âCalculates☺the☺beginning☺of☺the☺month date.p0

p1â[<expD>] optional date; defaults to current date.p0

p1Examples:âlStartDate☺:=☺DTBOM(☺CTOD(☺"10/15/90" ) )p0
p1âÜ╪&& lStartDate will contain 10/01/90p0

p1Returns:☺âBeginning☺DATE☺of☺the☺MONTH☺in which the passed parameter falls;p0
âp1NIL if invalid parameter is passed.p0
```

[illegible]

p1Function: DTBOY([<expD>]) --> datep0

p1Synopsis: Calculates the beginning of the year a given date falls in.p0

p1â[<expD>] optional date; defaults to current date.p0

p1Examples: lStartYear := DTBOY(CTOD("10/15/90"))p0
p1âÜ╬&& lStartYear will contain 01/01/90p0

p1Returns: Beginning DATE of the YEAR in which the passed parameter falls;p0
âp1NIL if invalid parameter is passed.p0

DTCHKDTEFMT()

p,Function: DtChkDteFmt(<expC>) --> arrayp0

p1Synopsis: Accepts a dateformat string and checks its validity.p0

p1â<expC1> Date formatstring as used by the SET() functionp0

p1âComparison is done according to the country specific date formatsp0
âp1as defined in CLIPPER when using the SET(_SET_DATEFORMAT)p0
âp1function.p0

p1âcountry identifiers as defined by CLIPPER:p0

p1âÜ1 = AMERICANp0
p1âÜ2 = ANSIp0
p1âÜ3 = BRITISHp0
p1âÜ4 = FRENCHp0
p1âÜ5 = GERMANp0
p1âÜ6 = ITALIANp0
p1âÜ7 = JAPANp0
p1âÜ8 = USAp0

p1Returns: an array with 2 elementsp0

p1âelement[1]contains .T. if dateformat was correct, otherwise .F.p0
âp1element[2]contains the country identifier as defined by CLIPPER ifp0
âp1dateformat was correct, otherwise 0p0

VERTEX CLIPPER Library:

All rights are reserved.

and this document are Copyright 1992 by W.A. Steinert,

```
DTDAYSIN()

Function: DTDAYSIN( <expD> ) --> integer

Synopsis: Calculates number of days in a month

p1â<expD>   a date which is used to calculate the number of dates inp0
âp1Ü╪the month the date is falling in.p0

Examples: ? DAYSIN( CTOD( "02/29/1992" ) )
p1âÜ╪&& prints the number 29p0

Returns: an integer representing the number of days of the month ap0
âp1passed Date falls into. 0 if an error occured.p0
```

```
K1x1MÖ╢VERTEX☺CLIPPER☺L☺☺IBRARY☺0.9☺USER'S☺GUIDE☺☺PAGE☺55☺☺OF☺182Px0☺☺☺☺☺☺☺☺☺☺☺☺☺☺

█CK1x1MÖ╢DTDIFF()☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺

K1Function:âDTDIFF(☺<expD1>,☺<expD2>,☺<expN>☺) --> integerp0

K1Synopsis:âcalculates☺the☺difference☺between☺two given dates depending uponp0
âp1action chosen.p0

p1â<expD1> start datep0

p1â<expD2> end datep0

p1â<expN>  action as defined in VERTEX.CH:p0

x1MâÜPx0p1DT_DAY= in daysp0
x1MâÜPx0p1DT_WEEK  = in weeksp0
x1MâÜPx0p1DT_MONTH = in monthsp0
x1MâÜPx0p1DT_YEAR  = in yearsp0

K1Examples:âlDate1☺☺☺:=☺CTOD(☺"10/15/90"☺)p0☺☺☺☺☺
p1âlDate2  := CTOD( "10/26/90" )p0
p1âdiffdays  := DTDIFF(adate,adate2, DT_DAY)p0
p1âdiffweeks := DTDIFF(adate,adate2, DT_WEEK)p0
p1âdiffmonth := DTDIFF(adate,adate2, DT_MONTH)p0
p1âdiffyears := DTDIFF(adate,adate2, DT_YEAR)p0

K1Returns:☺âinteger☺representing☺the☺difference of both dates depending uponp0
âp1action chosen.p0
```

DTDINM()

p1Function:âDTDINM([<expD>]) --> integerp0

p1Synopsis:âCalculates number of days in month of given date.p0

p1â[<expD>] optional date; defaults to DATE()p0

p1âThis function also takes consideration of Leap Years.p0

p1Examples:âDTDINM(CTOD("10/20/89")) returns 31p0

p1Returns:âan integer representing the amount of days in the month ofp0
âp1parameter or current date.p0

```
█♥K╟FDTDTOF()♦H¶↕☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺

K1Function:âDTDTOF()☺☺-->☺stringp0☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺

K1Synopsis:âReturns☺the☺current☺Date-Format☺stringp0

p1âThe function returns the current date setting as a format stringp0
âp18 characters OR 6 characters long, unlike CLIPPERS DTOS()p0
âp1function which will always return a string YYYYMMDD.p0

K1Examples:âSET☺DATE☺TO☺ANSIp0☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺
p1âlFormat := DTDTOF()p0

p1âÜ&& lFormat contains the string "yyyy.mm.dd" if centuryp0
p1âÜ&& is set on, otherwise it contains "yy.mm.dd"p0

K1Returns:☺âa☺date☺format☺string☺according to the current settingp0
```

DTDTOW()

```
p,Function:âDTDTOW( [<expD>] ) --> stringp0

p1Synopsis:âConverts a date to a sentence.p0

p1âThe Function convertes a given date to a sentence.p0

p1â[<expD>] optional date; defaults to DATE()p0

p1Examples:âDTDTOW( CTOD( "12/10/90" ) ) && returns "December 10th, 1990"p0
p1âDTDTOW( CTOD( "12/01/90" ) ) && returns "December 01st, 1990"p0
p1âDTDTOW( CTOD( "12/03/90" ) ) && returns "December 03rd, 1990"p0

p1Returns:âa string containing a date converted to a sentence if correct datep0
âp1was passed; otherwise an empty stringp0
```

```
p1x1MÖ╫VERTEX CLIPPER LIBRARY 0.9 USER'S GUIDE  PAGE 59 OF 182px0☺☺☺☺☺☺☺☺☺☺☺☺☺☺

■♥K╟F┼X()JN¶↕☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺

p,Function:âDTDOY( <expD> ) --> integerp0☺☺☺☺☺☺

p1Synopsis:âCalculates the day-of-year from a given date.p0

p1â<expD>  a date; defaults to current datep0

p1Examples:âDTDOY( CTOD("03/17/92") ) && returns 77p0

p(Returns:  âan integerp0☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺
```

```
DTSETDATE()

p1Function: DTSETDATE( [<expN1>] | [<expC1>] ) --> stringp0

p1Synopsis: Sets a date to a defined date formatp0

p1â[<expN1>]│[<expC1>]ßnumeric or string value identifying the wantedp0
âp1date formatp0

p1âWhen using a numeric value, keep in mind that only the followingp0
âp1values are correct:p0

p1âÜ╫1 = AMERICANp0
p1âÜ╫2 = ANSIp0
p1âÜ╫3 = BRITISHp0
p1âÜ╫4 = FRENCHp0
p1âÜ╫5 = GERMANp0
p1âÜ╫6 = ITALIANp0
p1âÜ╫7 = JAPANp0
p1âÜ╫8 = USAp0

p1âThe string values have been pre-defined in the headerfilep0
âp1VERTEX.CH and may used as followsp0

p1âÜ╫_AMERICANp0
p1âÜ╫_ANSIp0
p1âÜ╫_BRITISHp0
p1âÜ╫_FRENCHp0
p1âÜ╫_GERMANp0
p1âÜ╫_ITALIANp0
p1âÜ╫_JAPANp0
p1âÜ╫_USAp0

p1âYou may also define Your own dateformat however keep in mindp0
âp1that it should be somehow in the format:p0

p1âÜ╫╓dd:mm:yy·if century is set OFFp0
p1âÜ╫or╓dd:mm:yyyy   if century is set ONp0

p1âThe delimiters are being used the way You have defined them.p0

p1Notes: If CENTURY is set ON and You post a dateformat showing nop0
âp1century information, the function will not post your date. Thep0
âp1same goes if You set CENTRY OFF and want to post a dateformatp0
âp1which includes century information.p0

p1Returns: the current dateformat if new dateformat was posted successfully,p0
âp1otherwise empty stringp0
```

[illegible]

p1Function: DTWOY([<expD>]) --> integerp0

p1Synopsis: Calculates the week-of-year from a given date.p0

p1â[<expD>] a date; defaults to current datep0

p1Examples: DTWOY(CTOD("11/04/91")) && returns 45p0

p1Returns: an integerp0

IMPORTANT FILE FUNCTION INFORMATION

p1Not much to say about the FILE functions except again, all Functions are writt
p1CLIPPER 5.01 (unless otherwise marked) and not in beefed up SUMMER '87 (par
p1the pun).p0

p1Although the SUMMER '87 compiler was quite good for its time, there were impor
p1features missing which had to be programmed in a round about way (I think all
p1CLIPPER pro's agree with me). Now the 5.01 version does a better job.p0

VERTEX CLIPPER Library and this document are Copyright 1992 by W.A. Steinert,
All rights are reserved.

FIASKFILENAME()

p1Function:âFIAskFilename([<expC>], [<expN1>], [<expN2>..<expN3>]) --> stringp

p1Synopsis:âAsks the user for a filename. The function checks the validity ofp0
âp1the fielname to comply with DOS. The Function Key F2 allows thep0
âp1user to create a valid, unique Filename. Warns the user if filenamep0
âp1alreay exists.p0

p1â[<expC>] optional box colorstring; defaults to c_popcolp0

p1â[<expN1>] optional box shadow position; defaults to c_shadposp0

p1â[<expN2>..<expN3>]ßoptional box coordinates; default to screenp0
âp1center.p0

p1Examples:âFIASKFILENAME() returns a valid Filenamep0

p1Notes:âThe function p0p1

DOES☺NOT☺CREATE☺A☺FILE☺p0p1,☺☺it supplies only thep0
âp1filename.p0

p1(Returns:☺âa☺string☺containing☺a☺valid☺filename.p0

```
K1╤FiBldDbf()H¶☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺

p,Function:âFiBldDbf(☺<expC1>,<expC2>│<expA>☺)p0

p1Synopsis:âCreates☺a☺DBF☺from☺a☺delimited☺stringp0

p1â<expC1> name of DBF to createp0

p1â<expC2> long string delimited ...p0

p1âFields must be delimited with colons, the Field elements withp0
âp1commasp0

p1â--- or ---p0

p1â<expA> an array of Fields and Field elements delimited withp0
âp1commatasp0

p1Examples:âwhen☺<expC2>☺is☺a☺string...p0☺☺☺☺☺☺☺

p1âFiBldDbf( "CUSTOMER", "FNAME,C,30:ADDR,C,30:ZIP,N,10:CREDIT,L" )p0

p1âÜ&& creates a database called CUSTOMER with 4 Fieldsp0
p1âÜ&& FNAME... Type C, Length 30;p0
p1âÜ&& ADDR ... Type C, Length 30;p0
p1âÜ&& ZIP  ... Type N, Length 10;p0
p1âÜ&& CREDIT  ... Type L, Length 1p0

p1âwhen <expA> is used...p0

p1âlDbStruc := {╓"FNAME,C,30",;p0
p1âÜ╪╓"ADDR,C,30",;p0
p1âÜ╪╓"ZIP,N,10",;p0
p1âÜ╪╓"CREDIT,L" }p0

p1âFiBldDbf( "CUSTOMER", lDbStruc )p0

p1âdoes the same.p0

p1Notes:âFiBldDbf()☺expects☺an unused area to work in, and will return .f.p0
âp1if it detects a DBF open in the current area. An overwrite will notp0
âp1be allowed.p0

p1Returns:â.T.☺if☺database☺created☺successfullyp0
p1âotherwise .F.p0
```

FILEINFO()

p1Function: FILEINFO(<expC>, <expN>) --> stringp0

p1Synopsis: Examines a file and returns file info based upon passedp0
âp1parameters.p0

p1â<expC> filename of file to checkp0

p1â<expN>   what to look for. These options are definedp0
p1âÜin the p0p1

```
VERTEXUS.CHFp0p1 file and are as follows:p0

p1âÜ█FI_SIZEp0
p1âÜ█FI_DATEp0
p1âÜ█FI_TIMEp0
p1âÜ█FI_ATTRp0

K1Examples:
p1    a filesize = fileinfo("customer.dbf", FI_SIZE)p0
p1âÜ&& filesize contains the size of file "customer.dbf" asp0
p1âÜ&& a stringp0

K1Returns:
p1(   a string of the wanted fileinfop0
```

FIKEEPVAR()

p1Function:âFiKeepVar(<expC1>) --> Filenamep0

p1Synopsis:âWrites <expC1> to disk and returns a unique Filename.p0

p1â<expC1> a string of any kind.p0

p1âThis Function is very useful in writing saved screen variables top0
âp1disk for keeping them there for the duration of the program sop0
âp1they may be recalled at a later time. Allows saving quite somep0
âp1Memory.p0

p1âMEMO Fields may be exported into a file for editing with anp0
âp1external editor.p0

p1âJust remember to erase the Files. Do this by using the functionp0
âp1fitempclean() as last functioncall in your program.p0

p1Examples:âlScrSave := SAVESCREEN(0,0,20,50)p0
p1âlFilename := FiKeepVar(lScrSave)p0
p1âÜ&& lFilename contains a unique filename where thep0
âp1SAVESCREENp0
p1âÜ&& string was written to...p0

p1â- or -p0

p1âlFilename := FiKeepVar("this string is written to disk")p0

p1Returns:âFilename if successfully saved, otherwise empty stringp0

MAKEUNIQUE()

Function: MakeUnique([<expC1>]) ---> Filenamep0

Synopsis: Creates a unique Filename. Checks if Filename already exists andp0
âp1changes Filename accordingly.p0

p1â<expC1> optional extensionp0

p1âIf the optional extension is used the Filename construct is:p0

p1âÜ╪TMP#####.___p0

p1âwhere the underscores (___) in the Filename are replaced byp0
âp1the extension. Otherwise the filename defaults to the Filenamep0
âp1Construct:p0

p1âÜ╪TMP#####.##$p0

p1âwhere # denotes an integer in the range 0 .. 9p0

Notes: Use only valid extension characters for the filename as nop0
âp1checking on validity is done.p0

Returns: a unique filename not existing in default directory.p0

FITEMPCLEAN()

Function: FiTempClean() --> NIL

Synopsis: Erases all temporary files created by the user. Requires thep0
âp1PUBLIC array c_tfiles to be available where the names of allp0
âp1temporary files are stored in, e.g. with Functionp0

x1MâÜPx0p1fikeepvar()p0
x1MâÜPx0p1DBLOOKFOR()p0

Notes: Erases only the temporary files created by a specific user allp0
âp1others are left unchanged.p0

p1âCall this function as the very last thing you do before leaving ap0
âp1program since it will erase also the files in which evtl. Screenp0
âp1areas have been saved, e.g. with Functions:p0

p1âDBLOOKFOR()p0
p1âfikeepvar()╒etc.p0

p1âAlso, use the fitempclean() function only if these above functionsp0
âp1had been used previously or you wrote manually to the arrayp0
âp1c_tfiles.p0

Returns: Nothing

31VERTEX CLIPPER Library:↕and this document are Copyright 1992 by W.A. Steinert,

All rights are reserved.

FIVARGET()

```
Function: FIVarGet( <expC1> ) --> variable

Synopsis: Reads a temporary file in which a variable was saved.p0

p1â<expC1>  contains the filename into which the variable wasp0
âp1previously save with thep0
âp1Ü function FIKEEPVAR()p0

Notice: make sure that saved screen variable was read back correctlyp0
âp1otherwise may corrupt display.p0

Returns: Saved screen if successfully read, otherwise NULL valuep0
```

IMPORTANT GRAPHICS FUNCTION INFORMATION

p1I have supplied a view Graphics functions as a start. In general You do not ne
p1link anything special in to use them. However be aware that You do need a Grap
p1Video Adapter to use these functions.p0

p1Also, You must switch into a Graphics mode to be able to see the output and BE
p1your use any graphics functions.p0

p1CLIPPERS ? and ?? functions don't work in Graphics mode as well as the DEBUGGE
p1I will try and expand the Graphics Library a little in time to give You morep0
p1possibilities and will improve all functions so that You may use them in both
p1and GRAPHICS modes.p0

GRBOX()

p,Function: GRBOX(<expN1>..<expN5>) --> integerp0

p-Synopsis: Draws a Graphics box on screenp0

p1âThe function will draw a box on a screen in graphics mode a thep0
âp1given coordinates and returns the sum of a pixels drawn.p0

p1â<expN1>	TOP LEFT X coordinatep0
p1â<expN2>	TOP LEFT Y coordinatep0
p1â<expN3>	BOTTOM RIGHT X coordinatep0
p1â<expN4>	BOTTOM RIGHT Y coordinatep0
p1â<expN5>	COLOR of boxp0

p-Examples: GRBOX(100,100,600,400,3)p0

p1Notes: Remember You are in graphics mode so all coordinates are inp0
âp1PIXELS and NOT IN COLUMNS AND ROWS.p0

p1âIn general:p0

p1âÜ╪standard CGA has 320 x 200p0
p1âÜ╪standard EGA has 640 x 350p0
p1âÜand╪standard VGA has 640 x 480 pixels.p0

p(Returns: integer, number of pixels drawn; 0 on errorp0

█ GRINIT()

K1Function:âGRINIT(<exprN1>, <exprN2>) --> NILp0

K1Synopsis:âSets maximal X,Y coordinates for Graphics Screensp0

p1âCall this function whenever You want to set the maximal x,yp0
âp1screen coordinates to any other values than the ones set by thep0
âp1UTINIT() function.p0

p1âThe globals b_maxx and b_maxy are set according to whatp0
âp1Your system allows at maximum.p0

p1âIf Your video card supports 800 x 600 pixels, call this functionp0
âp1with:p0

p1âGRINIT(800, 600)p0

p1âsetting the maximal x,y coordinates to this value to benefit fromp0
âp1the provided Graphics functionsp0

K1Examples:âGRINIT(800,600) && set maximal x,y coordinates toà☺800x600p0
p1âGRINIT(640,480) && ·"►☺"☺"☺☺☺640x480p0

K1Returns:âNothingp0

```
█♥K╫K╪MO()H¶↕☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺

p,Function:âGRLINE(☺<expN1>☺☺,☺<expN5>☺☺)☺--->☺integerp0

p1Synopsis:âDraw☺a☺colored☺line☺in☺Graphics☺mode.p0

p1â<expN1>  starting X coordinatep0

p1â<expN2>  starting Y coordinatep0

p1â<expN3>  ending X coordinatep0

âp1<expN4>  ending Y coordinatep0

p1â<expN5>  color of linep0

p1âThe function will draw a line on the current display page fromp0
âp1starting X,Y coordinates to ending X,Y coordinates in the specifiedp0
âp1color.p0

p1Notes:âIf☺starting☺X,Y☺and☺ending X,Y are the same coordinates only onep0
âp1pixel is plotted.p0

p(Returns:âinteger,☺the☺number☺of☺actual☺pixels plotted.p0
```

GRPTREAD()

p1Function:âGRPTREAD(<expN1>, <expN2>) --> integerp0

p1Synopsis:âReads a colored pixel at X,Y coordinate and returns the palettep0
âp1color at a given point by *pposp0

p1â<expN1> X coordinate of the point whose color to read and returnp0

p1â<expN2> Y coordinatep0

p1Examples:âlColor = GRPTREAD(600, 300)p0
p1âÜ&& reads color at position 600,300 andp0
p1âÜ&& returns a numerical colorp0

p1Returns:âintegerp0

GRPTWRITE()

```
p1Function:âGRPTWRITE( <expN1> , <expN3> ) --> NILp0

p1Synopsis:âWrites a colored pixel at X,Y coordinatep0

p1â<expN1>  X coordinatep0
p1â<expN2>  Y coordinatep0

p1â<expN3>  color of pixelp0

p1âThe function will write a "colored" pixel at a given X,Y coordinate.p0

p1Notes:âMake sure that the pixel is within the bounds of the validp0
âp1Screenarea ( see b_maxx and b_maxy coordinates ) which You setp0
âp1with UTINIT() to default values or using the GRINIT() function dop0
âp1specifiy Your own.p0

p1Examples:âGRPTWRITE( 100,200,3 )p0

p1Returns:âNothingp0
```

IMPORTANT META FUNCTION INFORMATION

p1The META Functions are something special in many ways. I have them included fo
p1the beginners (or the overworked experts) to have a way to quickly put toget
p1program with all needed features to satisfy the customer.p0

p1A META Function is a function which may stand on itself and supply a complete
p1of functionality to the user (i.e. the MFDATAENTRY() function) and could qui
p1be a program on its own.p0

p1Use them if You want to, but be aware that they take away some space.p0

VERTEX CLIPPER Library and this document are Copyright 1992 by W.A. Steinert,

All rights are reserved.

MFCOLORPICK()

p1Function:âMFCOLORPICK() --> NILp0

p1Synopsis:âA meta-function allowing the selection of colors from a listp0

p1âThe Function gives the user the option via a menu to select atp0
âp1present 12 pre-defined color combinations which will be saved inp0
âp1the file SYSTEM.DATp0

p1âColors will be initialised with the SETCOLOR() functionp0

p1Returns:âNothingp0

T↑VERTEX CLIPPER Library↕and this document are Copyright 1992 by W.A. Steinert,
All rights are reserved.

MFDBTOOLS()

```
Function: MFDbTools([<expN1>,<expN2>,[<expA1>],[<expN3>],[<expC2>],[<expLp0
âp12>]) --> NILp0

Synopsis: A Database Toolbox containing major database handling functions.p0
âp1These functions can be disabled for use by means of the thirdp0
âp1parameter which requires an array of logical values.p0

p1â<expN1> Coordinate, top row of toolbox windowp0
p1â<expN2> Coordinate, left side of toolbox windowp0
p1â[<expA1>] ╓optional array of logical values denoting if item isp0
âp1selectable or not:p0

p1âÜ╫( .T. = selectable )p0
p1âÜ╫( .F. = not selectable )p0

p1âdefaults to ALL selectable.p0

p1â[<expN3>] optional parameter of windows shadowp0

p1â[<expC2>] optional parameter of window colorp0

p1â[<expL2>] optional parameter for display windows with or w/op0
âp1framep0

p1âFollowing functions are available with this toolbox:p0

p1âÜ1. Export data (in part or whole )p0
âp1Ü2. Import datap0
âp1Ü3. Shrink database ( removes deleted records )p0
âp1Ü4. Jump database ( selects new working database )p0
âp1Ü5. Zap database ( removes ALL data from database )p0
âp1Ü6. Interactive build index and set if desiredp0
âp1Ü7. Print selected recordsp0
âp1Ü8. Change program parametersp0

Examples: MFDBTOOLS( 5,5,{.T.,.T.,.T.,.T.,.T.,.T.,.T.,.T.} )p0

p1âÜ╫╓&& pops up the tools and all being enabledp0

p1âMFDBTOOLS( 5,5 )ß&& the same....p0

Returns: Nothingp0
```

```
K1x1MÖ╣VERTEX☺CLIPPER☺L☺☺IBRARY☺0.9☺USER'S☺GUIDE☺☺PAGE☺79☺☺F☺182Px0☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺



╬♥K╟MFDATAENTRY()H¶↕☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺

K1Function:âMfDataEntry(☺[<expA1>☺☺,☺<expA3>]☺) --> NILp0

K1Synopsis:âAllows☺generic☺Data-manipulation.☺Requires an open database andp0
âp1three passed arrays. If no parameters are passed, then defaultsp0
âp1to all fields being displayed and the FIELDNAMES being the fieldp0
âp1descriptors. No indexes are then being used.p0

p1â<expA1>  an array containing the Database-Structure of thep0
âp1database currently in use.p0
âp1Ü╬Use the function DBSTRUCT() to get the required infop0
âp1about the structure,p0
âp1Ü╬e.g.p0

p1âÜ╬╔lStruct := DBSTRUCT()p0

p1âÜ╬and pass the array "lStruct" as first param.p0

p1â<expA2>  an array of field descriptors of the current Database.p0

p1â<expA3>  an array of logical values. Any .T. value results in thep0
âp1field beingp0
âp1Ü╬displayed, .F. prevents the field being displayed.p0

K1Example:☺âUSE☺samplep0☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺
p1âlStruc  := DBSTRUCT()·►☺☺☺☺&& Structurep0
p1âlFields := {"Field1","Field2","Field3"}☺☺☺&& Fielddescriptorsp0
p1âlYN  := { .T., .F., .T. }  ►☺☺☺☺&& display ? YES/NOp0
p1âlNtx := { "NAME","ACCOUNT" }    ☺☺&& 2 indexes are usedp0
p1âMfDataEntry( lStruc, lFields, lYN, lNtx )☺☺&& functioncallp0

p1âResults in FIELDS  "Field1" and "Field3" being displayed andp0
âp1possible to be edited. "Field2" will not appear.p0


p1âUSE samplep0
p1âMfDataEntry()p0

p1âResults in ALL fields being displayed with their FIELDNAMESp0
âp1being the descriptors.p0

K1Notice:☺âAll☺Functionkeys☺in☺the range F1 - F10 being SET with thep0
âp1SETKEY() CLIPPER Function will be saved and returned to theirp0
âp1former settings. Requires a database in use in the currentp0
âp1workareap0

K1Returns:☺âNothingp0☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺
```

```
¤K╫MAKEEXPR()H¶☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺

K1Function:âMfMakeExpr(☺<expA>,☺[<expC1>],☺[<expN1>] ) --> Expressionp0

K1Synopsis:âAllows☺the☺interactive☺creation☺of☺an expression.p0

p1âThis expression can be used to find certain data in a database,p0
âp1which matches the expression. An array of fieldnames must bep0
âp1passed. There is no checking if the fieldnames passed, fit thep0
âp1database in use. So make sure that the database is in use ofp0
âp1which the fieldnames have been passed in the array <expA>p0

p1â<expC1> optional colorstringp0

p1â<expN1> optional shadowpositionp0

p1âThe screen will be build in a fashion of:p0
â$â┌──box1──────┤ ├ ┌box4x2─────┤ ├ ┌──box3─────┤↕┘J$J§

p1âwhere the boxes 1 to 4 following options contain:p0

p1âBox1:╫all fields selectable by the user.p0
âp1Box2:╫operator used for the comparisonp0
âp1Box3:╫operand used for the comparison, which can bep0
âp1Üa) a value already existing in the databasep0
p1âÜb) a new value to be entered by the user.p0

p1âalternatively contains .T. or .F. option when selecting a field ofp0
âp1type LOGICAL.p0

p1âBox4: the spot where the new value can be entered.p0

p1âNote that box 4 is only relevant if the selected field is NOT ofp0
âp1type "L" (logic).p0

p1âFields of type "M" (Memo) cannot be included in the expression.p0

K1Returns:☺a☺valid☺expression☺if☺completed, an empty string if unsuccessfulp0
```

J

MFPICKCOLOR()

Function: MFPickColor() --> logical value

Synopsis: pops up a selection of color choices and stores it to filep0
âp1SYSTEM.DATp0

Returns: .T. if successfully changed, .F. if no changes were madep0

```
K1x1MÖ╫VERTEX☺CLIPPER☺L☺☺IBRARY☺0.9☺USER'S☺GUIDE☺☺PAGE☺82☺☺OF☺182Px0☺☺☺☺☺☺☺☺☺☺☺☺☺☺



#♥K╫SELFILE()H¶↕☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺

K1Function:âMfSelfile(☺[<expC>]☺)☺☺->☺stringp0

K1Synopsis:âA☺META☺function☺for☺selecting☺a☺PATH/File. Pops up a window inp0
âp1the center of the screen allowing the user to select either a file,p0
âp1or change a directory permanently.p0

p1â[<expC>]╬optional three character extension to search for.p0
p1âÜ╬possible extensions currently are:p0

p1âÜ╬╔ALLp0
âp1Ü╬╔DBFp0
âp1Ü╬╔NTXp0
âp1Ü╬╔FRMp0
âp1Ü╬╔SDFp0
âp1Ü╬╔ASCp0

p1âothers will be added soon.p0

p1âI have tried to put something together the quick way for Yourp0
âp1convenience to change a directory and select a file. I know it'sp0
âp1not the ideal thing but please be free to make suggestions.p0

p1âA row of buttons on the right side allow You to select the type ofp0
âp1files You want to see or select in the left window. Choose thep0
âp1Filetype, then choose the file. Push the "RETURN" button to havep0
âp1the full filename ( w/o the path returned ).p0

K1Examples:âlFile☺:=☺MFSELFILE()☺►☺&&☺to☺return a filenamep0

p1âlFile := MFSELFILE( "DBF" )►☺&& allows selection of DBF filesp0
p1âÜ╬╔ß·&& ONLY. Other options such asp0
p1âÜ╬╔ß·&& DIR change etc. are not affectedp0

K1Returns:âa☺string☺containing☺the☺filenamep0
```

```
MFSETCOLOR()

p1Function:âMFSETCOLOR() --> NILp0

p1Synopsis:âDevelopment Tool for finding color combinations.p0

p1âSets the global color- and display type variables.p0

p1âSeveral global (public) color variables are used by the functions.p0
âp1These are:p0

p1âc_normcol╫-╔For normal input/outputp0
p1âc_normmenu╔-ßFor normal 'menu to' operationsp0
p1âc_popcol╫-╔For popup box colorsp0
p1âc_popmenu╔-ßFor popup box menusp0
p1âc_errcol╫-╔For popup error boxesp0
p1âc_frame╫-╔Frame stringp0
p1âc_shadatt╫-╔Shadow color attribute (numeric)p0
p1âc_shadpos╔-ßShadow position (0,1,3,7,9)p0
p1âc_xplode╫-╔Logical - explode windows?p0
p1âp0
p1âAll are of the format "f/b,f/b,,,f/b" (f-foreground b-background)p0

p1âMFSETCOLOR() allows interactive setting of these colors. Thep0
âp1variables are stored in SYSTEM.DAT - which is created if neededp0
âp1by MFSETCOLOR().p0

p1âIf SYSTEM.DAT is not present, UTINIT() will initialise a default setp0
âp1of colors, otherwise it will restore from SYSTEM.DATp0


p1Notes:âMFSETCOLOR() does not effect the current color setting. Itp0
âp1manipulates SYSTEM.DAT. To utilize colors selected byp0
âp1MFSETCOLOR(), a restore from SYSTEM.DAT must be done.p0

p1Returns:âNothingp0
```

8♥K╠1╠IMPORTANT☺MENU☺FUNCTION☺INFORMATION¶↕☺☺☺

p1The Menu functions make use of other VERTEX-functions as well as several low l
p1C and Assembler functions. They are fast, small and easy to use.p0

p1Currently I have stayed away from SAA type menus as I found them using quite s
p1memory and processing overhead.p0

```
#♥K╟MNABORT()¶↕

p1Function:âMNABORT( [<expC>], [<expN1>..<expN2>] ) --> logicalp0

p1Synopsis:âPops up a dialog box asking the user "ABORT?" and allows top0
âp1make a choice if an escape key ( ASCII 27 ) was pressed at a waitp0
âp1state.p0

p1â<expC>   optional characterstring of the box color defaults to thep0
âp1globaly definedp0
âp1Ü╫POPMENU colorp0

p1â<expN1> .. <expN2>ßoptional box position where. expN1 is thep0
âp1equivalent of TOPp0
âp1Ü╫╔ßROW and expN2 is the equivalent of LEFTp0
âp1COLUMNp0

p1Example:âINKEY(0)p0
p1âIF MNABORT()p0
p1âÜEXITp0
p1âENDIFp0

p1Returns:â.T. for yes, .F. for nop0
```

MNALERT()

p1Function: MNALERT(<expC1>, <expA1>) --> integerp0

p1Synopsis: A popup dialog box with more than 2 choices displayed centered.p0

p1â<expC1> Expression to be displayed as question.p0

p1â<expA1> answersp0

```
p1Examples: lChoice := mnalert( "Are you ready", 10, 10, { "Yes!", "Not yet",p0
âp1"Almost" } )p0
p1âDO CASEp0
p1âÜCASE lChoice == 1  && yesp0
p1âÜ... your code ...p0

p1âÜCASE lChoice == 2  && not yetp0
p1âÜ... your code ...p0

p1âÜCASE lChoice == 3  && almostp0
p1âÜ... your code ...p0

p1âÜOTHERWISE && ESCp0
p1âENDCASEp0
```

p1Returns: an integer identifying the selected option; 0 on error or if nop0
âp1selectionp0

VERTEX CLIPPER Library:

MNCHOICE()

```
p1Function:âMNCHOICE( <expA>, <expN1>..<expN4>, [<expC1>], [<expL>],p0
âp1[<expC2>], [<expN5>] ) --> integerp0

p1Synopsis:âThe equivalent of CLIPPER's ACHOICE() function however it alsop0
âp1creates a box for it with an optional heading and restores thep0
âp1underlying screen after selection.p0

p1â<expA>   an array of choices.p0

p1â<expN1>..<expN4>location of choice box.p0

p1â[<expC1>] optional boxtitlep0

p1â[<expL>] optional execute flag; defaults to .F. - if set to .T., thisp0
âp1function willp0
âp1Ü execute a return on a first letter match.p0

p1â[<expC2>] optional boxcolor; defaults to C_POPCOLp0

p1â[<expN5>] optional boxshadow position; defaults to C_SHADPOSp0

p1Examples:âMyArray := { "Apples", "Pears", "Bananas" }p0
p1âlChoice := MNCHOICE( MyArray,Top,Left,Bottom,Right,"Fruits",.T.)p0

p1Returns:âan integer identifying the selected array element.p0
p1â0 for no selectionp0
```

MNFIELDS()

```
p1Function:âMNFIELDS(╓[<expC1>],[<expN1>..<expN4>], [<expL>],p0
âp1[<expC2>],[<expN5>] ) --> string│integerp0

p1Synopsis:âpops up a box to allow selection of fields of current database.p0

p1â[<expC1>] ╓optional title of box.p0

p1â[<expN1>..<expN4>]ßoptional box coordinates; default to screenp0
âp1center.p0

p1â[<expL>]╪optional returnvalue selector. If set to .T. returnsp0
âp1selected FIELDNAME,p0
âp1Ü╪if set to .F. returns the  FIELDPOSITION of the selectedp0
âp1field;p0
âp1Ü╪defaults to .T.p0

p1â[<expC2>] ╓optional box colorstring; defaults to c_popcolp0

p1â[<expN5>] ╓optional box shadowposition; defaults to c_shadposp0

p1Notes:âDo not confuse this function with the MNTAKEFIELD() function.p0
âp1MNFIELDS() examines the current database and returns thep0
âp1fieldname or the fieldposition depending upon the 5th parameterp0
âp1set, whereas MNTAKEFIELD() relies upon the fieldnames passed asp0
âp1an array and does not need an open database.p0

p1Returns:âif parameter 5 is set to .T. --> selected fieldname if parameter 5p0
âp1is set to .F. --> FIELDPOSITION within current database ofp0
âp1selected Fieldp0
```

[illegible]

p1Function:âMNINFO(<expC1>,<expN1>,<expA1>) --> stringp0

p1Synopsis:âDisplays a popup box with up to as many lines of a messages,p0
âp1that fit on the screen, NOT waiting for a Keypress.p0

p1âThe amount that will fit on the screen will be calculated out of:p0

p1âÜmaximum number of displayable rows - 6p0

p1âThe maximum width that will be displayed is:p0

p1âÜmaximum number of displayable columns - 6p0

p1âThe message remains until removed with UNBOX()p0

p1â<expC1> message box color string; defaults to C_POPCOLp0

p1â<expN1> position of the box-shadow; defaults to C_SHADPOSp0

p1â<expA1> an array of messagesp0

p1Examples:âMNMSG(,,{ "An error has been detected!" })p0
p1âMNMSG(,,{ "A","B","C","D","E","F" })p0

p1Notes:âThe message is centered on screen in a box.p0

p1Returns:âunderlying screen in string.p0
p1âNothing on error.p0

```
█♥K1MENU()♠H¶↕☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺

K1Function:âMNMENU(☺<expC1>☺,☺<expC3>,☺[<expC4>☺☺.<expC16>]) --> integerp0

K1Synopsis:âA☺vertical☺popup☺menu☺in☺a☺centered box using <expC1> as titlep0
âp1and performing a MENU TO ... on <expC2> to <expC16> wherep0
âp1<expC1> may be passed as empty string for a popup menu w/out ap0
âp1title.p0

K1Examples:â1Choice☺☺:=☺mnmenu("Selection",☺"Edit","Add","Delete","Quit")p0

K1Returns:☺âan☺integer☺representing☺the☺position of the selected item. 0 (zero)p
âp1of nothing was selected.p0
```

MNMSG()

```
K1Function:âMNMSG(<expC1>,<expN1>,<expC2>│<expN2>,[<expC3>..<expC10>])p0
âp1Ü--> NILp0

K1Synopsis:âDisplays a popup box with up to 9 lines of a message, waitingp0
âp1then for a Keypress or optionally a time-out.p0

p1â<expC1> message box color string; defaults to C_POPCOLp0

p1â<expN1> message box shadow position; defaults to 3 (bottom rightp0
âp1)p0

p1â<expC2>│<expN1>p0
âp1a message string or a numeric value. If this parameter is ap0
âp1messagestring, than the message will be displayed as the firstp0
âp1message line within the message box; however if this parameter isp0
âp1of a numeric value, than the function will see it as the optionalp0
âp1timeout value after which the box will be automatically removedp0
âp1from the screen - otherwise the box will remain until a furtherp0
âp1key is pressed.p0

p1â[<exp3>..<expC10>] optional further message strings.p0

K1Examples:âMNMSG(,,"An error has been detected!")p0
p1âMNMSG(,,"A","B","C","D","E","F")p0
p1âMNMSG(,,5,"A","B","C","D","E","F")  && removed after 5 secondsp0

K1Notes:âThe message is centered on screen in a box. Once the message isp0
âp1displayed, the function waits for a keypress and then removes thep0
âp1window, restoring screen underneath, or waits for time-out valuep0
âp1defined as 3rd parameter.p0

K1Returns:âNothingp0
```

MESSYN()

p1Function: MESSYN(<expC1>, [<expC2>,<expC3>]) --> logicalp0

p1Synopsis: A popup YESNO dialog box.p0

p1â<expC1> Expression to be displayed as question.p0

p1â[<expC2>,<expC3>] optional answers; defaults to YES/NOp0

p1Examples: if messyn("Are you ready")p0
p1âif messyn("Are you ready","Not Yet","Almost")p0

p1Returns: .T. if first answer (leftmost) was selected,p0
p1â.F. if right answer (rightmost) was selected.p0

VERTEX CLIPPER Library and this document are Copyright 1992 by W.A. Steinert,

All rights are reserved.

```
K1╫MNPULLDN()♦☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺

K1Function:☺âMNPULLDN(☺<expN1>,☺<expA1>☺..☺<expA3> ) --> floatp0

K1Synopsis:☺âPulldown☺Menu☺creator☺and☺handler.☺☺Draws a pulldown menu withp0
âp1up to 8 menu boxes. selection process starts with position passedp0
âp1with <expN1>.p0

p1â<expN1>  a floating point number representing the start of thep0
âp1menu process, i.e. 1.01p0
âp1Ü╫starts with 1st Menu, 1st Menu optionp0

p1â<expA1> .. <expA3>  three arraysp0

p1â<expA1>  each element defines the Menu. A Menu definition is ap0
âp1String delimitedp0
âp1Ü╫with colons for each Menuoption.p0

p1âExample: ╫"TITLE:option:option:option..."p0

p1â Ü╫with up to 99 options per string.p0

p1â<expA2>  is a box definition array. It is 7 elements long andp0
âp1defines the following:p0

p1âElement:╫1. <expL> Draw top bar box? T/Fp0
p1âÜ╫2. <expC> Top bar colorp0
p1âÜ╫3. <expC> Menu Box colorp0
p1âÜ╫4. <expC> Menu Box framep0
p1âÜ╫5. <expN> Menu Box shadow position (0,1,3,7,9)p0
p1âÜ╫6. <expN> Menu Box shadow attributep0
p1âÜ╫7. <expN> Row to start menu barp0

p1â<expA3>  is an array of title column positions to override thep0
âp1internal 'figerinp0
âp1Ü╫algorithm.p0

K1Examples:☺â#include☺"vertex.ch"p0☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺

p1âBox := {╫"Datafiles:Use Datafile",;p0
p1âÜ╫"Indices:Select Indices:Index order",;p0
p1âÜ╫"Editing:Field Replacement:Tabular Edit",;p0
p1âÜ╫"Reporting:Build Query:Print Lists",;p0
p1âÜ╫"Other:List text file:Change Directory",;p0
p1âÜ╫"Quit" }p0

p1âBoxSel := 1.01p0

p1âBoxData := {╔.T.,;ß·►☺☺☺&& draw the top bar boxp0
p1âÜ╫╔"W/B,GR/R,,,W/N",;☺☺&& top bar color stringp0
p1âÜ╫╔"W/B,N/R,,,W/N",;☺☺&& drop box colorp0
p1âÜ╫╔VERTEX_SSSS,; ☺☺☺&& drop box framep0
p1âÜ╫╔3,;  ß·►☺☺☺&& drop box shadowpositionp0
```

```
p1âÜ╪╓8,;  ß·►☺☺☺&& drop boxp0
âp1shadowattributep0
p1âÜ╪╓0 }  ·►☺☺☺☺&& row # of menu barp0
p1âDO WHILE .T.p0
p1âÜBoxSel := MNPULLDN( BoxSel, Box, BoxData )p0
‼VERTEX CLIPPER Library↕and this document are Copyright 1992 by W.A. Steinert,
All rights are reserved.
```

```
J♥p1âÜDO CASEp0
p1âÜ╬case BoxSel == 1.01 && Datafiles -> Use...p0
p1âÜ╬╓.......p0
p1âÜ╬case BoxSel == 2.01 && Indices -> Select...p0
p1âÜ╬case BoxSel == 2.02 && Indices -> Index order...p0
p1âÜ╬case BoxSel == 3.01p0
p1âÜ╬case BoxSel == 5.01p0
p1âÜENDCASEp0
p1âENDDOp0
```

J♥p1âPassing a menu array element with title only will result in nop0
âp1menu box, with the title being the only selection and returning ap0
âp1selection value of <MENU>.1. (i.e. a 'Quit' box)p0

p1Warnings: It takes some work to start all options and titles with a f
âp1letter that is unique so that first letter selection may take place.p0

p1Returns: a float number to identify relative position of chosen menu-optionp0
âp1where the whole numerical part identifies the master menu andp0
âp1the fraction the sub menu.p0

```
MNTAKEFIELD()

p1Function:âMnTakeField(<expA>,[<expN1>..<expN4>],[<expC>],[<expN5>],p0
âp1Ü╫╔[<expL>]) --> Fieldnamep0

p1Synopsis:âAccepts an Array of Fieldnames and displays it for serselection atp0
âp1coordinates <expN1> to <expN4>.p0

p1â<expA>   array of fieldnamesp0

p1â[<expN1>..<expN4>] optional coordinates of box; default is centerp0

p1â[<expC>]╫optional box colorstring; defaults to C_POPCOLp0

p1â[<expN5>]╫optional box shadow position; defaults to C_SHADPOSp0

p1â[<expL>]╫optional execute flag; defaults to .F. - if set to .T., thisp0
âp1function willp0
âp1Ü╫execute a return on a first letter match.p0

p1Examples:âlFields := { "Name", "Address", "City" }p0
p1âlChoice := MNTAKEFIELD( lFields, top, left, bott, right )p0

p1â- or -p0

p1âlFields := { "Name", "Address", "City" }p0
p1âlChoice := MNTAKEFIELD(lFields, top, left, bott, right,;p0
p1âÜ╫╔ßM->color, M->shadow, .T.)p0


p1Returns:âFIELDNAME if one was chosen,p0
p1âEmpty string if none was chosen.p0
```

IMPORTANT MATH FUNCTION INFORMATION

p1The Math functions provided are using wherever available a Math-Coprocessor.p0

p1Some functions are however simply adding elements which make use of codeblocks
p1speed up the adding process. These however don't use a Co-Processor.p0

p1Functions which are using a Co-Processor (whenever there is one available) hav
p1been identified as such.p0

MTARAVERAGE()

p1Function: âMTARAVERAGE(<expA1>) --> floatp0

p1Synopsis: âGets the average of all numbers in an array. There can be anyp0
âp1types of elements within the array as the function only calculatesp0
âp1the average of elements which are of type NUMERIC.p0

p1â<expA1> the array to scanp0

p1Examples: âlArray := { "just for fun", 12, 200, 11.6, 97.12, .T. }p0
p1âlResult := MTARAVERAGE(lArray)p0

p1âÜ&& lResult contains 80.18 as averagep0

p1Returns: âa float containing the average of all NUMERIC elementsp0
p1â0 if no NUMERIC elements found or on Error.p0

```
MTARSUM()

p1Function:âMTARSUM( <expA1> ) --> floatp0

p1Synopsis:âSums a numeric array.p0

p1â<expA1> numerical array to sump0

p1Examples:âlArray := { 10, 11, 1, 2 }p0
p1âlTotal := MTARSUM( lArray ) && lTotal contains 24p0

p1Returns:âa float containing the sum of the array passed,p0
p1âon Error NILp0
```

```
VERTEX CLIPPER LIBRARY 0.9 USER'S GUIDE PAGE 100

K1M182Px0
MTDBAVERAGE()

p1Function:âMTDBAVERAGE( <expC1>|<expN1>, <expN2> ) --> floatp0

p1Synopsis:âGets the average of all numbers in a Field identified by <expC1>p0
âp1or <expN1>.p0

p1âExpects an open database in the current Workarea.p0

p1â<expC1>  the name of the field to average orp0

p1â<expN1>  the fieldnumber to averagep0

p1â<expN2>  Type 0 =  averages all records incl. deleted,p0
p1âÜ╪Type 1 =  averages only deleted records,p0
p1âÜ╪Type 2 =  averages all BUT deleted records.p0

p1Examples:âlResult := MTDBAVERAGE( 1, 0 )p0
p1âÜ&& lResult contains the average of Field 1 ALL recordsp0

p1âlResult := MTDBAVERAGE( "INCOME", 1 )p0
p1âÜ&& lResult contains the average of the Field "INCOME"p0
p1âÜ&& only deleted recordsp0

p1âlResult := MTDBAVERAGE( "PAYMENTS", 2 )p0
p1âÜ&& lResult contains the average of the Field "INCOME"p0
p1âÜ&& all BUT deleted recordsp0

p1Returns:âa float containing the average of the Fieldp0
p1â0 if no values in Database or on Error.p0
```

```
K1M182Px0
MTDBFLDSUM()

p1Function:âMTDBFLDSUM( <expC1>|<expN1> ) --> floatp0

p1Synopsis:âSum a Field of a database currently in use.p0

p1âThe function is adding up all values of a database field eitherp0
âp1identified by NAME or FIELDPOSITION.p0

p1â<expC1>  name of databasefield to sump0

p1â<expN1>  fieldposition of databasefield to sump0


p1Examples:âUSE ACCOUNTSp0
p1âlTotal := MTDBFLDSUM( 2 )p0

p1â - or -p0

p1âlTotal := MTDBFLDSUM( "PAYMENT" )p0

p1âboth do the same and return the same value.p0

p1Returns: âa float number representing the TOTAL of the Field.p0
p1â0 on errorp0
```

```
K⌠☺1MÖ╚VERTEX☺CLIPPER☺LIBRARY☺0.9☺USER'S☺GUIDE☺☺PAGE☺102☺☺FP☺0☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺

╜☺K1M182P☺0☺☺☺☺☺☺☺☺☺
╪CLIMTDEC2HEX()H¶↕☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺

p1Function:âMTDEC2HEX(☺<expN1>☺)☺--->☺stringp0☺

p1Synopsis:âConverts an Integer into a hexadecimal string.p0

p1âThe function accepts an integer and converts it into a hexadecimalp0
âp1string .p0

p1â<expN1> Integer to convertp0

p1Returns:âa string representing the integer, empty string if wrongp0
âp1parameter was passed.p0
```

```
K1MVERTEX CLIPPER LIBRARY 0.9 USER'S GUIDE  PAGE 103  FPx0

K1M182Px0
MTHEX2DEC()

p1Function:âMTHEX2DEC( <expC1> ) --> integerp0

p1Synopsis:âConverts hexadecimal string into an Integer.p0

p1âThe function accepts a hexadecimal string and converts it into anp0
âp1integer.p0

p1â<expC1> hexadecimal string to convertp0

p1Notes:âDon't pass the "H" which usually is used to identify a HEXp0
âp1number.p0

p1Examples:âMTHEX2DEC( "29" )  && returns 41p0
p1âMTHEX2DEC( "29h" )    && returns NILp0

p1âMTHEX2DEC( "A0A" )·&& returns 2570p0

p1Returns:âan integer equivalent of the hexadecimal string, NIL if wrongp0
âp1parameter was passed.p0
```

K1M182P0

IMPORTANT INDEX FUNCTION INFORMATION

p1The Indexfunctions are primarily used to provide an easy interface to create a
p1handle CLIPPER and DBASE indexes.p0

p1Some functions have the functionality of META functions and give the user quit
p1some control over the happenings.p0

182

NXISVALID()

```
p,Function:NXisvalid(<exp>) --> logicalp0

p1Synopsis:Checks if index expression is valid for database in currentp0
âp1workareap0

p1âRequires an open Database in the current workarea.p0

p1Returns: .T. if index expression is valid,p0
p1âotherwise .F.p0
```

```
VERTEX CLIPPER LIBRARY 0.9 USER'S GUIDE  PAGE 106

NXINDEXKEY()

p,Function:aNXINDEXKEY( <expC>, <expN> ) --> stringp0

p=Synopsis:aExtracts the key-expression of an index. Is sensitive to DBase IIIp0
âp1or Clipper generated indexes.p0

p1â<expC>   name of indexp0

p1â<expN>   index type. There are two different types of indexes (p0
âp1sofar ) usable withp0
âp1Ü╫CLIPPER:p0

p1âÜ╫╔ßType1 = CLIPPER NTX formatp0
p1âÜ╫╔ßType2 = DBase III NDX formatp0

p(Returns:aindex key-expression if sucessfully read,p0
p1âempty string if no expression found, i.e. invalid filep0
```

```
K⌡☺1MÖ╚VERTEX☺CLIPPER☺LI☺IBRARY☺0.9☺USER'S☺GUIDE☺☺PAGE☺107☺☺FPx0☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺

╜☺K1M182Px0☺☺☺☺☺☺☺☺
█GNXSCAN()H¶☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺

K1Function:âNxscan(☺[<expC>]☺)☺-->☺arrayp0☺☺☺☺☺

K1Synopsis:âScans☺a☺workarea☺for☺indexes.p0☺☺☺☺☺

p1â[<expC>] optional workarea name; defaults to current workarea.p0

p1âa workarea is scanned for open indexes. If a workarea is passedp0
âp1as parameter, it will be scanned for all open indexes, otherwisep0
âp1the current workarea will be scanned. All open indexes arep0
âp1written into an array.p0

K1Examples:âallindexes☺:=☺{}p0☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺
p1âallindexes := nxscan( ALIAS() ) && scans current workareap0

p1â-or-p0

p1âallindex := nxscan()&& scans current workareap0

p1â-or-p0

p1âallindex := nxscan( "NEWAREA" ) && scans workarea "NEWAREA"p0

p1â-or-p0

p1âAREA := "OTHERS"p0
p1âallindex := nxscan( AREA )&& scans workarea "OTHERS"p0

K1Returns:☺array☺containing☺open☺indexesp0
p1âempty array if no index foundp0
```

```
K⌡☺1MÖ╙VERTEX☺CLIPPER☺LI☺IBRARY☺0.9☺USER'S☺GUIDE☺☺PAGE☺108☺☺FP☺0☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺

╜☺K1M182P☺0☺☺☺☺☺☺☺☺
☺GK╤D╞SEEK()H¶↕☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺

p,Function:âNxDbSeek(☺[<expL>]☺)☺--->☺integerp0

p1Synopsis:âSeeks☺a☺record☺in☺the☺current☺workarea.p0

p1â[<expL>]╫optional setting for Softseek on or off; defaults to .F.p0
âp1where:p0

p1âÜ╫╒.F. = SOFTSEEK OFFp0
p1âÜ╫╒.T. = SOFTSEEK ONp0

p1âThe function will seek a record in the current workarea using thep0
âp1controlling index. Depending upon the Indexkey, the user isp0
âp1required to enter values of the fields containedin the index andp0
âp1in the sequence appearing in the Index.p0

p1Notes:âSoftseek☺is☺used,☺irrespective of overall settings, according top0
âp1[<expL>].p0

p1Examples:âINDEX☺ON☺SURNAME+FNAME☺TO☺NAMESp0☺☺
p1âSET INDEX TO NAMESp0
p1âlRecFound := NxDBSeek( .T. )p0

p1âÜ&& seeks a record using the index "NAMES" - Softseekp0
p1âÜ&& is set ON - the function requests input for FIELDSp0
p1âÜ&& SURNAME and FNAMEp0

p1Returns:☺âa☺recordnumber☺if☺SEEK☺was☺successful,p0
p1âotherwise 0p0
```

NxSelect()

p1Function:âNxSelect([<expN1>], [<expC>], [<expN2>]) --> logicalp0

p1Synopsis:âAllows the setting of indexes in the current workarea;p0

p1â[<expN1>]a numeric value of the controlling index (0 - 15)p0
âp1where 0 sets the controlling index off so that the database can bep0
âp1accessed in its natural order.p0

p1â[<expC>]optional colorsetting of selection box; default to c_popcolp0

p1â[<expN2>]option box-shadow position; default to c_shadposp0

p1âif the parameter <expN> is not passed, then the function will scanp0
âp1the current workarea for open indexes and allows interactivep0
âp1setting of the controlling index.p0

p1Notes:âRequires an open database in the current workareap0

p1Examples:âNxSelect(2)╓&& sets indexorder to 2p0

p1â- or -p0

p1âNxSelect()╓ß&& pops up a choice of available indexesp0
p1âÜ╓&& and allows interactive setting of thep0
p1âÜ╓&& indexorderp0

p1Returns:â.T. if successful, .F. if unsuccessfulp0

```
K⌡⊠1MÖ╙VERTEX☺CLIPPER☺LIBRARY☺0.9☺USER'S☺GUIDE☺☺PAGE☺110☺☺FP⊠0☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺

╜☺K1M182P⊠0☺☺☺☺☺☺☺☺☺
⊠IMPORTANT PRINT☺FUNCTION☺INFORMATION¶↕☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺

p1The provided printfunctions give You support in printing records of a database
p1making use of existing QUERY sets created with the MFMAKEEXPR() function.p0
```

```
╜☺K1M182P☺0☺☺☺☺☺☺☺☺
╘L│╔PROUTPUT()H¶↕☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺

p1Function:âPROutput( <expC> │ <expA> )  --> NILp0

p1Synopsis:âPrints the contents of a database while using a report formp0
âp1passed as parameter <expC>. Requires an open database in thep0
âp1current workarea.p0

p1âTo use this function best, copy first of all the records You wantp0
âp1to have included in the report into a seperate database and use itp0
âp1temporarily. This is the faster way of selecting records You wantp0
âp1to print.p0

p1âThe function will than give You a progressing report by means ofp0
âp1a counter on how many records still to go.p0

p1â<expC>   name of reportform to be used. If the DOS extender .FRMp0
âp1is not beingp0
âp1Ü╬used, it is assumed.p0

p1â- or -p0

p1â<expA>   array of setup information 10 elementsp0

p1âThe function requires the user to set certain parameters. Thesep0
âp1are:p0

p1âÜ╬a)  output to PRINTER or FILEp0
âp1Ü╬b)  WITHOUT or WITH ECHO while printingp0
âp1Ü╬c)  PLAIN or WITH HEADINGp0
âp1Ü╬d)  FIRST PAGE EJECTp0
âp1Ü╬e)  FULL REPORT or only SUMMARYp0

p1âAll these parameters may already be set by passing the requiredp0
âp1values together with the reportform name in an array of 10p0
âp1elements as the parameter. The Setup of this array should be:p0

p1âÜ╬{╓<expAC1>,;p0
âp1Ü╬╓<expAL1>,;p0
âp1Ü╬╓<expAL2>,;p0
âp1Ü╬╓<expAL3>,;p0
âp1Ü╬╓<expAL4>,;p0
âp1Ü╬╓<expAL5>,;p0
âp1Ü╬╓<expAL6>,;p0
âp1Ü╬╓<expAL7>,;p0
âp1Ü╬╓<expAL8>,;p0
âp1Ü╬╓<expAL9> }p0

p1âwhere the single elements represent following:p0

p1â<expAC1> ╓name of report formp0

p1â<expAL1> ╓.T. to Printer, .F. to File,;p0
```

p1â<expAL2> ╔.T. without Screen Echo, .F. with Screen Echo;p0

p1â<expAL3> ╔.T. Plain, .F. with Heading;p0

J   p1â<expAL4> ╔.T. first page eject, .F. no first page eject;p0

âp1<expAL5>╬.T. full report, .F. only summary;p0

p1â<expAC2>╬Heading string to be used if <expAL3> is set to .F.p0
âp1(printing withp0
âp1Ü╬heading). Note: Will be ignored if <expAL3> is set top0
âp1value .T.p0

```
K⌠☺1MÖ╚VERTEX☺CLIPPER☺LI☺IBRARY☺0.9☺USER'S☺GUIDE☺☺PAGE☺113☺☺Fp☺0☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺

╜☺K1M182p☺0☺☺☺☺☺☺☺☺☺☺

J       p1â[<expAN1>]╔an optional integer identifying the recordnumber ofp0
âp1a single recordp0
âp1Ü╫╔to be printed. Note: [<expAN2>] will be ignored ifp0
âp1anything is inp0
âp1Ü╫╔this element.p0

p1â[<expAN2>]╔an optional integer identifying the next "n" -p0
âp1amount of recordsp0
âp1Ü╫╔to be printed. Note: [<expAN1>] will be ignored ifp0
âp1anything is inp0
âp1Ü╫╔this element.p0

p1â[<expAL6>]╔optional logical value to determine if all records arep0
âp1to be processedp0
âp1Ü╫╔or only the REST ( from current position to end ofp0
âp1file ). .T. ifp0
âp1Ü╫╔printing the REST, .F. to print all; Note: Bothp0
âp1[<expAN1>] andp0
âp1Ü╫╔[<expAN2>] will be ignored if anything is in thisp0
âp1element.p0

p1Notes:âUsing only a formname when calling the function has the samep0
âp1result as using a parameter-array with following settings:p0

p1âÜ╫╔{ "< formname >",;☺&& <- Formnamep0
âp1Ü╫╔  .T.,;·►☺☺☺&& <- output to printerp0
âp1Ü╫╔  .T.,;·►☺☺☺&& <- no echop0
âp1Ü╫╔  .T.,;·►☺☺☺&& <- plainp0
âp1Ü╫╔  .T.,;·►☺☺☺&& <- eject first pagep0
âp1Ü╫╔  .T.,;·►☺☺☺&& <- full reportp0
âp1Ü╫╔  NIL; ·►☺☺&& <- headingp0
âp1Ü╫╔  NIL,;·►☺☺&& <- single recordnumber top0
âp1printp0
âp1Ü╫╔  NIL,;·►☺☺&& <- next n-records to printp0
âp1Ü╫╔  .F.,;·►☺☺☺&& <- print all recordsp0
âp1Ü╫╔  }p0

p1âMake sure all Elements which are not used contain a NIL value.p0

p1Examples:âproutput( "MYFORM" ) && prints using MYFORM.FRMp0
p1âÜ╫╔ß&& and defaultsp0

p1â- or -p0

p1â//--------------------------------------------------------p0
p1âÜThis array-setup will:p0

p1âÜa) use form "MYFORM.FRM"p0
âp1Üb) output to printerp0
âp1Üc) give no echo to screenp0
âp1Üd) uses default headings, thus ignoring the header writtenp0
âp1into element 7.p0
âp1Üe) eject the first pagep0
```

```
âp1Üf) do a FULL reportp0
âp1Üg) output all records in databasep0

p1âprnArray := {╓"MYFORM",    ☺&& <- Formnamep0
p1âÜ╬╓.T.,;ß·►☺☺&& <- output to printerp0
p1âÜ╬╓.T.,;ß·►☺☺&& <- no echop0
```

```
J       p1âÜ╪╔.T.,;ß·►☺☺&& <- plainp0
p1âÜ╪╔.T.,;ß·►☺☺&& <- eject first pagep0
p1âÜ╪╔.T.,;ß·►☺☺&& <- full reportp0
p1âÜ╪╔"Headings";►☺☺☺&& <- will be ignored thisp0
âp1time!!!!p0
p1âÜ╪╔NIL,;ß·►☺☺&& <- single recordnumber top0
âp1printp0
p1âÜ╪╔NIL,;ß·►☺☺&& <- next n-records to printp0
p1âÜ╪╔.F.,;ß·►☺☺&& <- print some or all recordsp0
p1âÜ╪╔ß} )p0
```

```
K⌠⊗1MÖ╚VERTEX☺CLIPPER☺LI☺IBRARY☺0.9☺USER'S☺GUIDE☺☺PAGE☺115☺☺FP⊗0☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺

╜☺K1M182P⊗0☺☺☺☺☺☺☺☺☺

J       p1â - or -p0

p1â//--------------------------------------------------------------p0
p1âÜThis array-setup will:p0

p1âÜa) use form "MYFORM.FRM"p0
âp1Üb) output to File. The Function will ask for a filename andp0
âp1will use it as outputp0
âp1Üfile.p0
âp1Üc) give no echo to screenp0
âp1Üd) uses the header written into element 7.p0
âp1Üe) eject the first pagep0
âp1Üf) do a FULL reportp0
âp1Üg) output all records in databasep0

p1âÜprnArray := {ß"MYFORM",    && <- Formnamep0
p1âÜ█╔ß.F.,;·►☺☺&& <- output to Filep0
p1âÜ█╔ß.T.,;·►☺☺&& <- no echop0
p1âÜ█╔ß.F.,;·►☺☺&& <- with headingp0
p1âÜ█╔ß.T.,;·►☺☺&& <- eject first pagep0
p1âÜ█╔ß.T.,;·►☺☺&& <- full reportp0
p1âÜ█╔ß"Headings";☺☺&& <- will be used thisp0
âp1time!!!!p0
p1âÜ█╔ßNIL,;·►☺☺&& <- single recordnumber top0
âp1printp0
p1âÜ█╔ßNIL,;·►☺☺&& <- next n-records to printp0
p1âÜ█╔ß.F.,;·►☺☺&& <- print some or all recordsp0
p1âÜ█╔ß·} )p0

p1â- or -p0

p1â//--------------------------------------------------------------p0
p1âÜThis array-setup will:p0

p1âÜa) use form "MYFORM.FRM"p0
âp1Üb) output to printerp0
âp1Üc) give no echo to screenp0
âp1Üd) uses default headings, thus ignoring the header writtenp0
âp1into element 7.p0
âp1Üe) eject the first pagep0
âp1Üf) do a FULL reportp0
âp1Üg) output ONLY the RECORD 1 as identified.p0

p1âÜprnArray := {ß"MYFORM",    && <- Formnamep0
p1âÜ█╔ß.T.,;·►☺☺&& <- output to printerp0
p1âÜ█╔ß.T.,;·►☺☺&& <- no echop0
p1âÜ█╔ß.T.,;·►☺☺&& <- plainp0
p1âÜ█╔ß.T.,;·►☺☺&& <- eject first pagep0
p1âÜ█╔ß.T.,;·►☺☺&& <- full reportp0
p1âÜ█╔ß"Headings";☺☺&& <- will be ignored thisp0
âp1time!!!!p0
p1âÜ█╔ß1,;  ·►☺☺&& <- single recordnumber top0
âp1printp0
```

```
p1âÜ╪╔ßNIL,;·►☺☺&& <- next n-records to printp0
p1âÜ╪╔ß.F.,;·►☺☺&& <- print some or all recordsp0
p1âÜ╪╔ß·} )p0
```

```
J       p1â- or -p0

p1â//---------------------------------------------------------p0
p1âÜThis array-setup will:p0

p1âÜa) use form "MYFORM.FRM"p0
âp1Üb) output to printerp0
âp1Üc) give no echo to screenp0
âp1Üd) uses default headings, thus ignoring the header writtenp0
âp1into element 7.p0
âp1Üe) eject the first pagep0
âp1Üf) do a FULL reportp0
âp1Üg) outputs the NEXT 200 RECORDSp0

p1âÜprnArray := { ·"MYFORM",  ☺&& <- Formnamep0
p1âÜ╪╓ß.T.,;·►☺☺&& <- output to printerp0
p1âÜ╪╓ß.T.,;·►☺☺&& <- no echop0
p1âÜ╪╓ß.T.,;·►☺☺&& <- plainp0
p1âÜ╪╓ß.T.,;·►☺☺&& <- eject first pagep0
p1âÜ╪╓ß.T.,;·►☺☺&& <- full reportp0
p1âÜ╪╓ß"Headings";☺☺&& <- will be ignored thisp0
âp1time!!!!p0
p1âÜ╪╓ßNIL,;·►☺☺&& <- single recordnumber top0
âp1printp0
p1âÜ╪╓ß200,;·►☺☺&& <- next n-records to printp0
p1âÜ╪╓ß.F.,;·►☺☺&& <- print some or all recordsp0
p1âÜ╪╓ß·} )p0
p1â- or -p0

p1â//---------------------------------------------------------p0
p1  âÜThis array-setup will:p0

p1âa) use form "MYFORM.FRM"p0
âp1b) output to printerp0
âp1c) give no echo to screenp0
âp1d) uses default headings, thus ignoring the header written intop0
âp1element 7.p0
âp1e) eject the first pagep0
âp1f) do a FULL reportp0
âp1g) output the REST of the records in the Database relative to thep0
âp1recordpointer position.p0

p1âprnArray := {╓"MYFORM",    && <- Formnamep0
p1âÜ╪╓.T.,;ß·►☺&& <- output to printerp0
p1âÜ╪╓.T.,;ß·►☺&& <- no echop0
p1âÜ╪╓.T.,;ß·►☺&& <- plainp0
p1âÜ╪╓.T.,;ß·►☺&& <- eject first pagep0
p1âÜ╪╓.T.,;ß·►☺&& <- full reportp0
p1âÜ╪╓"Headings";►☺☺&& <- will be ignored this time!!!!p0
p1âÜ╪╓NIL,;ß·►☺&& <- single recordnumber to printp0
p1âÜ╪╓NIL,;ß·►☺&& <- next n-records to printp0
p1âÜ╪╓.T.,;ß·►☺&& <- print the restp0
p1âÜ╪╓ß} )p0
```

p1âproutput(prnArray) && uses the defined array for parametersp0

p1(Returns .T. if print was successful, otherwise .F.p0

ZIVERTEX CLIPPER Library↕and this document are Copyright 1992 by W.A. Steinert,

All rights are reserved.

```
K⌠x1MÖ╚VERTEX☺CLIPPER☺LIBRARY☺0.9☺USER'S☺GUIDE☺☺PAGE☺117☺☺FPx0☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺

╜☺K1M182Px0☺☺☺☺☺☺☺☺☺
xIMPORTANT STRING☺FUNCTION☺INFORMATION¶↕☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺

p1Although CLIPPER provides String functions I have endeavored to provide You wi
p1some functions giving You more control over the handling.p0

p1Some functions make use of low level C functions for speeding things up. They
p1however all fully overlayable and are in many ways faster and smaller than the
p1provided CLIPPER string functions.p0
```

K1M182Px0

STARRANGE()

p1Function:âSTARRANGE(<expC>, <expN1>, <expN3>) --> stringp0

p1Synopsis:âRearranges a string.p0

p1â<expC> string to work on.p0
p1â<expN1> starting position for extractp0
p1â<expN2> length of extractp0
p1â<expN3> new startplace of extract AFTER text was extractedp0

p1Samples: âlString := "WINTERGARDEN"p0
p1âlString := STARRANGE(lString, 7, 6, 1)p0

p1â(returns "GARDENWINTER")p0

p1Returns: âaltered Stringp0

```
STBEGINSWITH()

Function: STBEGINSWITH( <expC1>, <expC2> ) --> logicalp0

Synopsis: Determines if a string begins with another stringp0

p1â<expC1> target string to comparep0

p1â<expC2> source string to compare target string withp0

Examples: lString1 := "MASTERPROGRAM"p0
p1âlString2 := "MASTER"p0
p1âlString2 := "SWITCH"p0

p1âSTBEGINSWITH( lString1, lString2 ) returns .T.p0
p1âSTBEGINSWITH( lString1, lString3 ) returns .F.p0

Returns: .T. if <expC1> begins with <expC2>, otherwise .F.p0
```

```
K1M182P0
STCENTER()

K1Function:âSTCENTER( <expC>, <expN> ) --> stringp0

K1Synopsis:âCenters a string within a given length. To do this, the functionp0
âp1will first pad the given string to the length required ( set withp0
âp1<expN> ) and centers the passed string ( in <expC> ) with thep0
âp1given length.p0

p1â<expC>  the string to be centered.p0

p1â<expN>  the length ( width ) in which string is to be centered.p0

K1Examples:âlNewString := STCENTER( "Old String", 20 )p0

p1âÜ&& lNewString is now 20 characters long and centeredp0
p1âÜ&& result:  "  Old String  "p0

K1Notes:âIf <expN> is less than the length of <expC>, the function willp0
âp1return the same string unaltered.p0

K1Returns:âa centered String if successful,p0
p1âthe passed string if unsuccessful,p0
p1âNIL if parameters are invalidp0
```

K1M182Px0

STCHRINSTR()

p1Function:âSTCHRINSTR(<expC1>, <expC2>) --> integerp0

p1Synopsis:âSearches a string for the first occurance of a character andp0
âp1returns its position if found.p0

p1â<expC1> string to be searchedp0

p1â<expC2> character to search forp0

p1Examples:âlString := "This is a string"p0

p1âlPos := STCHRINSTR(lString, "a")p0
p1âÜ&& lPos will contain 9p0

p1Returns:âan integer showing the position of the found character.p0

╜☺K1M182Px0☺☺☺☺☺☺☺☺

STCRUNCH()¶

K1Function:âSTCRUNCH(☺<expC>,☺<expN>☺)☺--->☺stringp0

K1Synopsis:âErases☺spaces☺contained☺in☺a☺string and appends them to the endp0
âp1of the string.p0

p1â<expC> string to work onp0

p1â<expN> action code as defined in VERTEXUS.CH:p0

p1âÜMOVE_ALLß·►☺= moves all spaces to end of stringp0
p1âÜMOVE_NO_SINGLE·= moves all but single spaces to end ofp0
âp1stringp0

K1Examples:âlString1☺:=☺"MOEVENPICK☺☺☺☺☺☺☺☺☺☺Ice Cream"p0

p1âSTCRUNCH(lString1, MOVE_NO_SINGLE)p0
p1âÜ╬&& returns "MOEVENPICK Ice Cream   "p0

p1â- or -p0

p1âSTCRUNCH(lString1, MOVE_ALL)p0
p1âÜ╬&& returns "MOEVENPICKIceCream   "p0

K1Returns:☺âcrunched☺string☺depending☺upon action, NIL if wrong parametersp0
âp1passedp0

```
K⌡☺1MÖ╙VERTEX☺CLIPPER☺LI☺IBRARY☺0.9☺USER'S☺GUIDE☺☺PAGE☺123☺☺FPx0☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺

╜☺K1M182Px0☺☺☺☺☺☺☺☺
╪LK╤☺STCUTLEFT()H¶↕☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺

K1Function:âstcutleft(☺<expC1>,☺<expN1>☺)☺——>☺stringp0

K1Synopsis:âThe☺function☺is☺cutting☺off☺the☺left side of a given string thep0
âp1amount of characters passed in <expN1>.p0

p1â<expC1> the string to cutp0

p1â<expN1> the amount of characters to cut offp0

K1Examples:âlString☺:=☺"A☺very☺long☺string☺we☺have!"p0
p1âlString := STCITLEFT( lString, 7 )p0

p1â&& lString = "long string we have!"p0

K1Returns:âremaining☺stringp0☺☺☺☺☺☺☺☺☺☺☺☺☺☺
```

STCUTRIGHT()

Function: stcutright(<expC1>, <expN1>) --> stringp0

Synopsis: The function is cutting off the right side of a given string thep0
âp1amount of characters passed in <expN1>.p0

p1â<expC1> the string to cutp0

p1â<expN1> the amount of characters to cut offp0

Example: lString := "A very long string we have!"p0
p1âlString := STCUTRIGHT(lString, 9)p0

p1â&& lString = "A very long string"p0

Returns: remaining stringp0

```
K1M182Px0
STENDSWITH()

K1Function:âSTENDSWITH( <expC1>, <expC2> ) --> logicalp0

K1Synopsis:âDetermines if a string ends with another stringp0

p1â<expC1> target string to comparep0

p1â<expC2> source string to compare target string withp0

K1Examples:âlString1 := "MASTERPROGRAM"p0
p1âlString2 := "PROGRAM"p0
p1âlString2 := "SWITCH"p0

p1âSTENDSWITH( lString1, lString2 ) returns .T.p0
p1âSTENDSWITH( lString1, lString3 ) returns .F.p0

K1Returns: â.T. if <expC1> ends with <expC2>, otherwise .F.p0
```

╜☺K1M182Px0☺☺☺☺☺☺☺☺
█1STLPAD()¶↕

p1Function:âSTLPAD(<expC>, <expN>) ---> stringp0

p1Synopsis:âPads a string with spaces at left side of stringp0

p1â<expC> string to padp0
p1â<expN> new length of stringp0

p1Notes:âstring will be truncated if <expN> is smaller than original stringp0
âp1length.p0

p1Returns:âstring left padded with spaces.p0

STLSHIFT()

p1Function: STLSHIFT(<expC>, <expN>) --> stringp0

p1Synopsis: Shift a string to left side. All leftmost characters are lost.p0

p1â<expC> string to shiftp0

p1â<expN> number of positions to shift leftp0

p1Returns: string as remained after left-shiftp0

STRPAD()

p1Function:âSTRPAD(<expC>, <expN>) --> stringp0

p1Synopsis:âPads a string with spaces at right side of stringp0

p1â<expC> string to padp0

p1â<expN> new length of stringp0

p1Notes:âstring will be truncated if <expN> is smaller than original stringp0
âp1length.p0

p1Returns:âstring right padded with spaces.p0

```
╜☺K1M182P☒0☺☺☺☺☺☺☺☺
☒LGK╒RSHIFT()H¶↕

p1Function:âSTRSHIFT( <expC>, <expN> ) --> stringp0

p1Synopsis:âShift a string to right side. All rightmost characters are lost.p0

p1â<expC>  string to shiftp0

p1â<expN>  number of positions to shift rightp0

p1Returns:âstring as remained after right-shiftp0
```

╜☺K1M182P⊠0☺☺☺☺☺☺☺☺☺☺

⊠1IMPORTANT☺SYSTEM☺FUNCTION☺INFORMATION¶↕☺☺☺

p1All System functions are low level C and Assembler functions. They are all wel
p1behaved and written in Microsoft C 6.00 and Turbo Assembler 1.0p0

p1I have sofar not experienced any problems with them and they should not give Y
p1any either. Never the less, since they go into the Hart of the Computer You sh
p1

use them cautiouslyFp0p1 and strictly the way suggested in the descriptions.p0

p1To speed things, I have ommitted any param checking in these functions as it i
p1assumed You know what You are doing.p0

```
K⌠⊗1MÖ╚VERTEX☺CLIPPER☺LIBRARY☺0.9☺USER'S☺GUIDE☺☺PAGE☺131☺☺FPx0☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺

╝☺K1M182Px0☺☺☺☺☺☺☺☺☺
╪1K╤☺H╪0╪♦)KH¶↕☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺

K1Function:âSYCHDIR(☺<expC1>☺)☺--☺>☺integerp0☺☺

K1Synopsis:âChanges☺the☺current☺directory.p0☺☺☺

p1â<expC1> the path where to change to.p0

K1Returns:â0☺if☺change☺occurredp0☺☺☺☺☺☺☺☺☺☺
p1âan integer < 0 on errorp0
```

```
K↑☺1MÖ╚VERTEX☺CLIPPER☺LIBRARY☺0.9☺USER'S☺GUIDE☺☺PAGE☺132☺☺FP☺0☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺

╜☺K1M182P☺0☺☺☺☺☺☺☺☺
╪GK╫YC╫HDRIVE()H¶↕☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺

K1Function:âSYCHDRIVE(☺<expC1>│<expN1>☺)☺-->☺logicalp0

K1Synopsis:âchanges☺the☺standard☺drive.p0☺☺☺☺☺☺☺

p1â<expC1>  a character identifying the drive to change top0

p1â<expN1>  a number identifying the drivenumber to change to thep0
âp1number representsp0
âp1Ü╬the drive as identified by DOS.p0

K1Returns:â.T.☺if☺successfulp0☺☺☺☺☺☺☺☺☺☺☺☺
p1â.F. on errorp0
```

SYCOLDBOOT()

Function: SYCOLDBOOT() --> NIL

Synopsis: Reboots the system (includes a RAM Check)

Returns: nothing

```
K⌠☒1MÖ╚VERTEX☺CLIPPER☺LIBRARY☺0.9☺USER'S☺GUIDE☺☺PAGE☺134☺☺FP☒0☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺

╜☺K1M182P☒0☺☺☺☺☺☺☺☺
☒1GSYCURDRV()H¶↕☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺

K1Function:âSYCURDRV()☺--> integerp0☺☺☺☺☺☺☺☺☺☺☺

K1Synopsis:âReturns☺the☺current☺Drivenumber.p0☺

p1âThe function checks the system and returns the current drive inp0
âp1form of a numberp0
âp1( 1 = A, 2 = B, 3 = C, ... etc. )p0

K1Returns:âan☺integer☺identifying☺the☺current drive.p0
```

T↑VERTEX CLIPPER Library↕and this document are Copyright 1992 by W.A. Steinert,

All rights are reserved.

```
K↑☺1MÖ╚VERTEX☺CLIPPER☺LI☺IBRARY☺0.9☺USER'S☺GUIDE☺☺PAGE☺135☺☺FP☺0☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺

╜☺K☺1M☺182P☺0☺☺☺☺☺☺☺☺
≠IGSYDRVEFREE()H¶↕☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺

p,Function:âSYDRVEFREE(☺<expN1>|<expC1>☺)☺-->☺integerp0

p1Synopsis:âChecks☺a☺drive☺and☺returns☺the☺free Diskspace.p0

p1â<expN1>|p0
p1â<expC1>  the identifier of the drive to be checkedp0

p1âif a number is passed it will represent the drive according to thep0
âp1DOS calling conventionp0

p(Returns:☺âa☺long☺integer☺representing☺the free diskspace available.p0
```

VERTEX CLIPPER Library↕and this document are Copyright 1992 by W.A. Steinert,

All rights are reserved.

SYDRIVE()

Function: SYDRIVE() --> character

Synopsis: Returns the current Drive.

p1âThe function is calling the C-Function SYCURDRV() to convert thep0
âp1drivenumber returned by the C-Function into the equivalentp0
âp1Character value.p0

Returns: a character identifying the current drive.p0

```
K⌡☺1MÖ╚VERTEX☺CLIPPER☺LIBRARY☺0.9☺USER'S☺GUIDE☺☺PAGE☺137☺☺FPx0☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺

╜☺K1M182Px0☺☺☺☺☺☺☺☺☺
x1G╫SYDRIVES()☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺

K1Function:aSYDRIVES()☺☺-->☺stringp0☺☺☺☺☺☺☺☺☺☺☺☺☺

K1Synopsis:aGet☺available☺drives☺of☺system☺an☺returns these in a string.p0


K#Return:astring☺of☺available☺drivesp0
```

```
K⌠☺1M☺VERTEX☺CLIPPER☺LI☺IBRARY☺0.9☺USER'S☺GUIDE☺☺PAGE☺138☺☺FP☺0

╜☺K☺1M182P☺0
SYDRVTTL()

p1Function:âSYDRVTTL(☺<expN1>|<expC1>☺)☺-->☺integerp0

p1Synopsis:âchecks☺a☺drive☺and☺returns☺the☺total Diskspace.p0

p1â<expN1>|p0
p1â<expC1>  the identifier of the drive to be checkedp0

p1âif a number is passed it will represent the drive according to thep0
âp1DOS calling conventionp0

p1Returns:âa☺long☺integer☺representing☺the total diskspace available.p0
```

```
K1M182Px0
SYDRVRDOK()

p,Function:âSYDRVRDOK( <expC1>|<expN1> ) --> logicalp0

p1Synopsis:âCheck if the drive, identified by the parameter is ready to readp0
âp1or not.p0

p1Examples:âIF SYDRVRDOK( "A" )p0
p1âÜ.... your code to read from disk ...p0
p1âELSEp0
p1âÜ.... other action since drive is not ready ....p0
p1âENDIFp0

p1(Returns:â.T. if drive is ready, otherwise .F.p0
```

```
╜☺K1M182Px0☺☺☺☺☺☺☺☺☺
xGLSYDRVWRTOK()H¶↕☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺

p,Function:âSYDRVWRTOK(☺<expC1>|<expN1>☺)☺──>☺logicalp0

p1Synopsis:âCheck☺if☺the☺drive,☺identified☺by☺the parameter is ready to writep0
âp1or not.p0

p1Examples:âIF☺SYDRVWRTOK(☺"A"☺)p0☺☺☺☺☺☺☺☺☺☺☺☺☺☺
p1âÜ.... your code to write to disk ...p0
p1âELSEp0
p1âÜ.... other action since drive is not ready ....p0
p1âENDIFp0

p(Returns:â.T.☺if☺drive☺is☺ready,☺otherwise .F.p0
```

```
VERTEX CLIPPER LIBRARY 0.9 USER'S GUIDE  PAGE 141

K1M182

SYMKDIR()

p1Function:âSYMKDIR( <expC1> ) --> integerp0

p1Synopsis:âCreates a subdirectory.p0

p1â<expC1> is string naming the new subdirectoryp0

p1Notes:â<expC1> should be a string such as You would use when creatingp0
âp1a subdirectory in the DOS environment with the DOS commandp0
âp1"MD" or "MKDIR".p0

p1Returns:â0 if subdirectory created successfully, any number less than 0 onp0
âp1error.p0
```

K1M182

SYRMDIR()

K1Function:âSYRMDIR(<expC1>) --> integerp0

K1Synopsis:âErases the subdirectory passed as parameter.p0

p1â<expC1> the subdirectory to be erased.p0

K1Notes:âThe subdirectory to be erased must be empty.p0

K1Returns:â0 if erased correctly, an integer < 0 on errorp0

K1M182P0

IMPORTANT GENERAL FUNCTIONS INFORMATION

p1In this category I have included all functions for which I have not establishe
p1group.p0

p1You may find some functions in here provided by someone else I have included t
p1here simply because they are good and complement the CLIPPER library in manyp0
p1ways. These functions were all distributed inclusive Source code, and I have i
p1the source code on the disk for Your usage.p0

p1Wherever I used someone elses function, I have made mentioning of it. I do not
p1to have any credit for it, nor do I want to make anything out of it. I simply
p1them and provide them to You for Your benefit.p0

```
K1M182Px0

UTBITSET()

K1Function:âUTBITSET( <expC1>, <expN1> ) --> character p0

K1Synopsis:âSets a selected bit in a BYTE (character)p0

p1â<expC1> character (BYTE) to set a bit inp0

p1â<expN1> bitposition to set ( 0 - 7, right to left )p0

p1âIn effect, ORs argument byte with a byte that has only the targetp0
âp1bit set.  If bit is already set, it remains set.p0

K1Examples:âNewbyte := UTBITSET( CHR(107), 4 )p0
p1â? ASC( Newbyte ) // result: 123p0
p1â? Newbyte  // result: '{'p0

K1Returns:âNew byte (character), with designated bit set.p0
p1âIf parameters are faulty, returns NIL.p0
```

UTBELLATTN()

Function: UtBellAttn() --> NIL

Synopsis: a single tone at 200 Hz for the duration of 1/18th of a second and 1100 Hz for the same duration.

Note: The tone sounds only if the global variable _quiet is set to .F.

Returns: nothing

```
UTBELLERROR()

Function: UtBellError() --> NILp0

Synopsis: a single tone at 100 Hz for the duration of 2/18th of a second.p0

Note: The tone sounds only if the global variable _quiet is set to .F.p0


Returns: nothingp0
```

UTBELLDONE()

Function: aUtBellDone() --> NIL

Synopsis: a single tone at 500 Hz for the duration of 1/18th of a second.

Note: The tone sounds only if the global variable _quiet is set to .F.

Returns: nothing

```
K1M182Px0

K1Function:âUtBellUdf( <expA1> ) --> NILp0

K1Synopsis:âa single tone or a combination of tones depending upon the valuesp0
âp1passed.p0

p1â<expA1> a 2-dim. array containing one or more tones.p0

p1âThe array should be created in following fashion:p0

p1âÜ{ { <frequency>,... }, { <duration>,... } }p0


K1Examples:âTo have the same effect with UtBellUdf() as with UtBellAttn()p0
âp1create an array like:p0

p1âÜAttnTone := { { 200, 100 }, { 1, 1 } }p0

p1âand call the function like:p0

p1âÜUtBellUdf( Attntone )p0


K1Note:âThe tone sounds only if the global variable _quiet is set to .F.p0


K1Returns:ânothingp0
```

```
K⌠☺1M☺ö╚VERTEX☺CLIPPER☺LI☺IBRARY☺0.9☺USER'S☺GUIDE☺☺PAGE☺149☺☺FPx0☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺

╜☺K1M182Px0☺☺☺☺☺☺☺☺
╪GK╫FP┼XBYTEAND()H¶↕☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺

K1Function:âUTBYTEAND(☺<expC1>,☺<expC2>☺)☺--->☺characterp0

K1Synopsis:âPerforms☺bit-wise☺AND☺on☺two☺BYTES☺(characters)p0

p1â<expC1> BYTE to be AND'edp0

p1â<expC2> to AND <expC1> withp0

p1â<expC1> and <expC2> are characters from CHR(0) TO CHR(255). Mayp0
âp1be passed in CHR() form, as character literals, or as expressionsp0
âp1evaluating to CHR() values.p0

p1âCan be used for any bit-wise masking operation. In effect, this isp0
âp1a bit-by-bit AND operation. Equivalent to AND assemblerp0
âp1instruction.p0

K1Returns:☺a☺resulting☺byte,☺in☺CHR()☺form. If parameters are faulty, returnsp0
âp1NIL.p0
```

```
K⌠☺1MÖ╚VERTEX☺CLIPPER☺LIBRARY☺0.9☺USER'S☺GUIDE☺☺PAGE☺150☺☺FP☺0☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺

╜☺K1M182P☺0☺☺☺☺☺☺☺☺☺
K1UTBYTEOR()H¶↕☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺

p,Function:âUTBYTEOR(☺<expC1>,☺<expC2>☺)☺-->☺characterp0

p1Synopsis:âPerforms☺bit-wise☺OR☺on☺two☺characters.p0

p1â<expC1> the character to be OR'dp0

p1â<expC2> the character to OR withp0

p1â<expC1> and <expC2> are characters from CHR(0) TO CHR(255). Mayp0
âp1be passed in CHR() form, as character literals, or as expressionsp0
âp1evaluating to CHR() values.p0


p1Examples:âThis☺code☺performs☺a☺bit-wise☺OR☺on two bytes represented byp0
âp1CHR(20) and CHR(10):p0

p1âNewChar := UTBYTEOR( CHR(20), CHR(10) )p0
p1â? ASC( NewChar ) ß// result: 30p0
p1â? NewChar  ╔ß// result: non-printable characterp0

p(Returns:☺âcharacter☺in☺CHR()☺form.☺☺If☺parameters are faulty, returns NIL.p0
```

```
K⌡x1MÖ╚VERTEX☺CLIPPER☺LI☺IBRARY☺0.9☺USER'S☺GUIDE☺☺PAGE☺151☺☺FPx0☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺

╜☺K1M182Px0☺☺☺☺☺☺☺☺☺
x1K╒☺UTCHKCURSTYPE()H¶↕☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺

p,Function:âUtChkCursType( <expN> ) ---> logical valuep0

p1Synopsis:âaccepts a numerical value representing the cursortype defined inp0
âp1CLIPPER for user with the SETCURSOR() functionp0

p1â<expN>  numerical expression as defined in SETCURS.CHp0

p(Returns: â.T. if numerical expression identifies valid cursorshape .F. ifp0
âp1numerical expression out of bounds or expression invalid.p0
```

```
UTCOLCONVERT()

p,Function:âUTCOLCONVERT( <expC1>, <expN1> ) --> integerp0

p1Synopsis:âreturns a color integer for colorsettingp0

p1âYou have Your colorstring but need the reference color integer.p0

p1â<expC1> colorstring to convertp0

p1â<expN1> what part of colorstring to retrieve as integer identifiedp0
âp1as:p0

p1âÜ╬0 = foreground standardp0
âp1Ü╬1 = background standardp0
âp1Ü╬2 = standard colorsettingp0
âp1Ü╬3 = foreground enhancedp0
âp1Ü╬4 = background enhancedp0
âp1Ü╬5 = enhanced colorsettingp0

p1Examples:âlColor := UTCOLCONVERT( SETCOLOR(), 2 )☺☺&& returns standardp0
p1âÜ╬╓ß·►☺☺☺☺&& color integerp0
p1âlColor := UTCOLCONVERT( SETCOLOR(), 2 )+128☺&& returns standardp0
p1âÜ╬╓ß·►☺☺☺☺&& color integerp0
p1âÜ╬╓ß·►☺☺☺☺&& and sets the blinkp0
p1âÜ╬╓ß·►☺☺☺☺&& attributep0

p1âlColor := UTCOLCONVERT( "W/N", 1 )☺☺☺&& returnsp0
âp1backgroundp0


p(Returns:âInteger representing the CLIPPER color number.p0
```

```
╜☺K1M182Px0
╪CLRF☺ROW(KH¶
p,Function:âUTCTRLW()p0
p1Synopsis:âSends a CONTROL-W ( CHR(23) ) to the keyboard.p0
p1âAllows remapping of a key to Ctrl-Wp0
p1Examples:âSET KEY F10 to UTCTRLWp0
p1â&& sends CTRL-W to keyboard whenever F10 is pressedp0
p1Returns:âNothingp0
```

```
K⌡☺1MÖ╚VERTEX☺CLIPPER☺LI☺IBRARY☺0.9☺USER'S☺GUIDE☺☺PAGE☺154☺☺FP☺0☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺

╜☺K1M182P☺0☺☺☺☺☺☺☺☺
╪GLK╔UTGETCURSTYPE()H¶↕☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺

K1Function:âUtGetCursType(ß[<expN1>..<expN4>], [<expC>], [<expN5>] )p0
p1âÜ╫╔--> integerp0

K1Synopsis:âPops up a window of selection of available cursor types.p0

p1âCall this function:p0

p1âÜcursshape := UtGetCursType()p0
p1âÜSETCURSOR( cursshape )p0

p1â[<expN1>..<expN4>]ßoptional window coordinates; defaults to screenp0
âp1center.p0

p1â[<expC>]╫optional box Colorstring; defaults to c_popcolp0
p1â[<expN5>]╫optional boxshadow position; defaults to c_shadposp0

p1âif no coordinates are given, the window is popped up in thep0
âp1center of the screen with default colors.p0

K1Returns:âan integer denoting the cursorshape to be used with the CLIPPERp0
âp1function SETCURSOR(). If no cursorshape was selected thep0
âp1function returns the currents cursor shape.p0
```

```
╜☺K1M182Px0☺☺☺☺☺☺☺☺
UTDRIVES()

K1Function:âUTDRIVES()☺☺☺☺>☺stringp0☺☺☺☺☺☺☺☺☺☺☺☺☺

K1Synopsis:âChecks☺the☺Computer☺System☺for☺available drives and returns infop0
âp1as string.p0


K1Examples:âlString☺:=☺UTDRIVES()p0☺☺☺☺☺☺☺☺☺☺☺☺☺☺

p1â&& lString may contain "ABC"p0
p1â&& if computer has 2 Floppies and 1 HDp0

K1Notes:âvirtual☺disks☺will☺be reportet as well This Function is a C-p0
âp1Function.p0
```

```
K↑☺1MÖ╚VERTEX☺CLIPPER☺LI☺IBRARY☺0.9☺USER'S☺GUIDE☺☺PAGE☺156☺☺FP☺0☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺

╜☺K1M182P☺0☺☺☺☺☺☺☺☺
#CUTEDPIC()¶↕☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺

p1Function:âUtedpic(☺<expC1>,☺<expN1>,☺<expN2>, [<expC2>] ) --> PICTUREp0

p1Synopsis:âReturns☺a☺picture☺clause☺appropriate for editing fieldp0

p1â<expC1> is the value of the field or variable.p0

p1â<expN1> denotes LEFTMOST COLUMN where GET is displayed.p0

p1â<expN2>  denotes RIGHTMOST COLUMN where GET is displayed.p0

p1â[<expC2>] allows optionally passing of userdefined PICTURE.p0

p1Notes:âChecks☺if☺RIGHTMOST☺column lies within Screen Boundaries andp0
âp1adjusts the column to POSSIBLE RIGHTMOST COLUMN - 2 if onlyp0
âp1the first 3 parameters are passed.p0

p1âBoundary checks are not done if [<expC2>] is passed as it isp0
âp1assumed that user knows what he is doing.p0

p1âIf Utedpic is used on Numeric fields, make sure that overallp0
âp1fieldlength does not exceed 15 positions EXLUSIVE the signs -+.()p0
âp1as they are counted as positions as well by CLIPPER, otherwisep0
âp1the number may not be displayed and Your will see the wholep0
âp1field displayed in default width filled with "*". Can't do anythingp0
âp1about this ( at least have not found a way around it ) as it seemsp0
âp1to be CLIPPER's own decision to do this.p0

p1Example:â@10,10☺GET☺FIRSTNAME☺PICTURE☺UTEDPIC( @FIRSTNAME, 10, 20 )p0

p1â&& would return a picture of "@KSXXXXXXXXXX"p0

p1Returns:☺âPicture☺-☺descriptionp0☺☺☺☺☺☺☺☺
```

```
K⌠☺1MÖ╚VERTEX☺CLIPPER☺LIBRARY☺0.9☺USER'S☺GUIDE☺☺PAGE☺157☺☺FPx0☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺

╜☺K1M182Px0☺☺☺☺☺☺☺☺
≈GK╞ENHANCED()H¶↕☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺

p,Function:âUTENHANCED()☺☺-->☺integerp0☺☺☺☺☺☺☺☺☺☺

p1Synopsis:âreturns☺a☺color☺integer☺for☺enhanced setting used for functionsp0
âp1that require it like ATT() and PRNT()p0

p1Examples:âlColor☺:=☺UTENHANCED()☺☺&&☺returns☺enhanced color integerp0
p1âPRNT(10,10,"Waiting...!",lColor )p0


p(Returns:☺âInteger☺representing☺the☺CLIPPER color number.p0
```

```
K⌡☺1MÖ╙VERTEX☺CLIPPER☺LI☺IBRARY☺0.9☺USER'S☺GUIDE☺☺PAGE☺158☺☺FP☺0☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺

╜☺K1M182P☺0☺☺☺☺☺☺☺☺
╪LK╒ERRORFUNCTION()H¶☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺

p,Function:âUtErrFunction(☺<exp0>☺)☺☺--->☺NILp0☺

p1Synopsis:âThe☺VERTEX☺Error☺handling☺function.p0

p1âHandles various kinds of errors and informs the user of the kindp0
âp1of error happening. Gives the user then a choice of actions whichp0
âp1will be stored in the ErrorObject:cargo slot for further action.p0

p1âAlthough no value is being returned, You are able to use thep0
âp1ErrorObject for Your RECOVERY.p0

p1âSeveral possible errors are being directly treated:p0

p1âÜ╫1)  opening a Databasep0
âp1Ü╫2)  opening an Indexp0
âp1Ü╫3)  DOS file open errorp0
âp1Ü╫4)  DOS file close errorp0
âp1Ü╫5)  File corruption errorp0

p1âE.g. You allow the user to open some file for usage but You cantp0
âp1be sure that the user opens a DBase Type file for reading. Thisp0
âp1would result in a fatal error crashing Your program. To have ap0
âp1chance to continue anyway (i.e. allow the user to reselect a file )p0
âp1do the following:p0

p1âÜ╫a) set up Your own error handlep0
âp1Ü╫b) allow the user to select the databasep0
âp1Ü╫c) open the database while using the statementp0

p1âÜ╫BEGIN SEQUENCE ...p0
âp1Ü╫RECOVER USING ...p0
âp1Ü╫ENDp0

p1âÜ╫d) test ErrorObject:cargo if recover is wanted.p0

p1Examples:âLOCAL lObjLocal,;☺☺☺☺☻☺☺☺&&☺contains the ErrorObjectp0
p1âlSelFile,;  ╔ß·►☺☺☺&& contains selectedp0
âp1filenamep0
p1âlSuccess := .T.╔ß·►☺☺&& set to false if file is badp0

p1âÜ╫╔ß·►☺☺&& install Errorfunctionp0
p1âlLastHandle := ERRORBLOCK( { | FileobjErr | UtErrFunction(p0
âp1FileobjErr ) } )p0

p1âDO WHILE .T.p0

p1âÜ.... allow the user to select a file .....p0

p1âÜBEGIN SEQUENCEp0
p1âÜ╫USE ( lSelFile ) NEW ☺☺&& use the selected filep0

p1âÜ╫RECOVER USING lObjLocal☺&& the ErrorObject will bep0
```

p1âÜ╫╔ß·►☺☺&& used here... Remember thatp0
p1âÜ╫IF ErrorObject:cargo ☺☺&& the CARGO slot willp0
âp1containp0

All rights are reserved.
VERTEX CLIPPER Library↕and this document are Copyright 1992 by W.A. Steinert,

```
╜☺K1M182P☺0☺☺☺☺☺☺☺☺☺☺

J       p1âÜ╪╔LOOP·►☺☺☺&& a .T. or .F. valuep0
âp1dependingp0
p1âÜ╪ELSEß·►☺☺☺&& on the action the userp0
âp1wantsp0
p1âÜ╪╔lSuccess := .F.►☺☺&& to do since his actionp0
p1âÜ╪╔EXITß·►☺☺&& failed.p0
p1âÜ╪ENDIFp0
p1âÜENDp0
p1âENDp0

K1(Returns:☺âNothingp0☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺
```

```
K↑x1MÖ╚VERTEX☺CLIPPER☺LIBRARY☺0.9☺USER'S☺GUIDE☺☺PAGE☺160☺☺FPx0☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺

╜☺K1M182Px0☺☺☺☺☺☺☺☺
╫UTEXPRISTRUE()╫¶↕☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺

p,Function:âUtExpristrue( <expC> ) --> logicalp0

p1Synopsis:âEvaluates an expression used against a Database record. Used byp0
âp1CLIPPER function DBEVAL() in second codeblock parameter if thep0
âp1expression is stored in a variable.p0

p1âE.g. lExpr := "SOUNDEX(FIELDGET(1)) = SOUNDEX("MASTERS")"p0

p1âTo use DBEVAL() correctly by using a variable as FOR or WHILEp0
âp1condition in the respective codeblock, the expression must bep0
âp1used as a MACRO to be evaluated to a LOGICAL value.p0

p1âE.g. DBEVAL( {|| <dowhatever if TRUE> }, {|| &lExpr } ).p0

p1âTo use an expression together with a macro in the secondp0
âp1codeblock, we have to call the UtExpristrue() function and passp0
âp1the expression to it for evaluation.p0

p1âE.g.ÜDBEVAL( {|| ... }, {|| firstfunc(),;p0
p1âÜUtExpristrue( lExpr ) } )p0

p1âThis runs the DBEVAL() function correctly.p0

p1Note:âIf you want to use this Expression Evaluation Function, make surep0
âp1it is called LAST within the second codeblock of the DBEVAL()p0
âp1function or things wont work.p0

p1âNo Typechecking is done to ensure maximum speed so make surep0
âp1you pass a valid expression. Best use the function MfMakeExpr()p0
âp1to create interactively a VALID expression.p0

p(Returns:â.T. if expression evaluates to .T. otherwise .F.p0
```

```
K↑☺1MÖ╚VERTEX☺CLIPPER☺LI☺IBRARY☺0.9☺USER'S☺GUIDE☺☺PAGE☺161☺☺FP☺0☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺

╜☺K1M182P☺0☺☺☺☺☺☺☺☺
≢L╫╤SAVEKEY()H¶↕☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺

p,Function:âUtSaveKey(☺<expL>☺)☺-->☺NILp0☺☺☺☺☺☺

p-Synopsis:âSaves☺and☺restores☺SET☺KEYS☺of☺F1☺to F10 keyp0

p1âIf you call this function, it will save or recall SET KEYSp0
âp1depending upon the parameter passed. This enables you to re-p0
âp1assign SET KEYS to functions after you have called this functionp0
âp1and RE-set the keys afterwards to their original settings.p0

p-Examples:âSET☺KEY☺K_F1☺to☺Help►☺&&☺sets☺F1☺to Help() functionp0
p1âUtSaveKey( .T. )ß·►☺&& save the settingp0
p1âSET KEY K_F1 to OtherHelp►☺&& sets F1 to OtherHelp() functionp0
p1âUtSaveKey( .F. )ß·►☺&& resets F1 to Help() functionp0

pLNotes:âDon't☺call☺the☺UtSaveKey() function twice in a row with thep0
âp1parameter .T. or the first saved functionkey setting are beingp0
âp1lost.p0

p1âe.g.ÜSET KEY K_F1 to Helpp0
p1âÜUtSaveKey( .T. )p0
p1âÜUtSaveKey( .T. )  <-- all settings are lost at this pointp0


p(Returns:â☺0☺if☺call☺was☺unsuccessfulp0☺☺
p1â 1 if functions are savedp0
p1â-1 if functions are restoredp0
```

K⌠x1MÖ╚VERTEX☺CLIPPER☺LI☺IBRARY☺0.9☺USER'S☺GUIDE☺☺PAGE☺162☺☺FPx0☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺

╜☺K1M182Px0☺☺☺☺☺☺☺☺
█GK╞UTINIT()H¶☺☺☺

K1Function:âUTINIT()☺☺--->☺NILp0☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺

K1Synopsis:âInitializes☺the☺global☺variables☺☺expected by functions located inp0
âp1the Library if they are not already present.p0

p1âIt will also look for a variable called _SETISCOLOR. If it isp0
âp1present and of type "L", its value will be used in place ofp0
âp1ISCOLOR(). You may want to initialize a public variablep0
âp1_SETISCOLOR if you have one of those card/monitor combos thatp0
âp1fool ISCOLOR().p0

p1âUTINIT() will also check Your system for the presence of ap0
âp1graphics adapter and set the graphics GLOBALS to the highestp0
âp1possible value in sequence of:p0

x1MâÜ╫Px0p1CGAp0x1M╓ßPx0p1EGAp0x1M·►☺Px0p1VGA resolutions.p0

p1âIf You have a graphics adapter with different resolutions than thep0
âp1standard listed above, You may use the function GRINIT() to setp0
âp1the maximal X,Y coordinates for Your system.p0

K1Examples:âat☺beginning☺of☺top☺level☺program.☺.p0

p1âUTINIT()p0

K1Notes:âIt☺always☺a☺good☺idea to call this at your top level program. Allp0
âp1of the functions doing screen i/o call it. It initializes the globalp0
âp1variable _CLIPFUNC the first time it is called.p0

K1Returns:âNothingp0☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺

```
K1M182Px0

p1Function:âUTISBIT( <expC1>, <expN1> ) --> logicalp0

p1Synopsis:âTest bitstatus of bits in a BYTE (character)p0

p1â<expC1> BYTE or character to examinep0

p1â<expN1>  Bitposition to examine (0 - 7, right to left )p0
p1âÜi.e.╓0 is equal to rightmost bit,p0
p1âÜ╓7 is equal to leftmost bit.p0

p1âChecks the identified Bit position in a given Byte for the statusp0
âp1of the identified Bit.p0

p1Examples:âlBitflag := UTISBIT(CHR(65), 7)p0

p1â? lBitflag// result: .T.p0

p1â? UTISBIT('A', 5)  // result: .F.p0

p1Returns:  â.T. if designated bit is set,p0
p1â.F. if bit is not set,p0
p1âotherwise NILp0
```

```
╜☺K1M182Px0☺☺☺☺☺☺☺☺
╪UTKEEPVAR()H¶↕☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺

p,Function:âUtKeepVar(☺<expC1>☺)☺--->☺Filenamep0

p=Synopsis:âWrites☺<expC1>☺to☺disk☺and☺returns☺a unique Filename.p0

p1â<expC1> a string of any kind.p0

p1âThis Function is very useful in writing saved screen variables top0
âp1disk for keeping them there for the duration of the program sop0
âp1they may be recalled at a later time. Allows saving quite somep0
âp1Memory.p0

p1âMEMO Fields may be exported into a file for editing with anp0
âp1external editor.p0

p1âJust remember to erase the Files. Do this by using the functionp0
âp1uttempclean() as last functioncall in your program:p0

p=Examples:âlScrSave☺:=☺SAVESCREEN(☺0,0,20,50☺)p0
p1âlFilename := UtKeepVar( lScrSave )p0
p1âÜ&& lFilename contains a unique filename where thep0
âp1SAVESCREENp0
p1âÜ&& string was written to...p0

p1â- or -p0

p1âlFilename := UtKeepVar( "this string is written to disk" )p0


p(Returns:âFilename☺if☺successfully☺saved,☺otherwise empty stringp0
```

K↑☺1MÖ╚VERTEX☺CLIPPER☺LI☺IBRARY☺0.9☺USER'S☺GUIDE☺☺PAGE☺165☺☺FPx0☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺

╜☺K1M182Px0☺☺☺☺☺☺☺☺
xp1KHTPROFILE()p0☺☺

K1Function:âutprofile()p0☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺

K1Synopsis:âThis☺is☺a☺Clipper☺5.01☺specific☺re-write of the profiler includedp0
âp1with Blinker 1.5x (originally written and donated by Frederick W.p0
âp1Stangl, President of Dynamic Performance Inc., Philadelphia, PA).p0

p1âThis version of the profiler was written by Todd C. MacDonald,p0
âp1Senior Programmer/Analyst of Key Services Corporation, Albany,p0
âp1NY, and is also donated to the Blinker community.p0

p1âThis version takes advantage of 5.01's nested arrays to allowp0
âp1viewing of the profiling statistics in any user-selectable order.p0

p1âTo invoke the profiler from any wait-state, press [Alt]-[P].p0

p1âI found this modul so good, I included in the library forp0
âp1development purposes.p0

p1âThe source-code may be ordered by sending me a SAE togetherp0
âp1with US $ 10.00 for postage and handling and I will provide Youp0
âp1with the source on disk; the code is also available on severalp0
âp1good Bulletin Boards and via COMPUSERVE in the Nantucket Userp0
âp1Forum.p0

K⌠⊗1MÖ╚VERTEX☺CLIPPER☺LI☺IBRARY☺0.9☺USER'S☺GUIDE☺☺PAGE☺166☺☺FP⊗0☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺

╜☺K1M182P⊗0☺☺☺☺☺☺☺☺
≢LK╟S╟UTSTANDARD()H¶↕☺☺

p,Function:âUTSTANDARD()☺--->☺integerp0☺☺☺☺☺☺☺☺☺

p1Synopsis:âreturns☺a☺color☺integer☺for☺standard setting used for functionsp0
âp1that require it like VIATT() and VIPRNT()p0

p1âThe functions writes the string direct to VIDEO MEMORY and isp0
âp1thus VERY FAST!!! However it doesn't work in all modes (somep0
âp1graphics modes). It works with all video modes using eitherp0
âp1B000h or B800h as base address (e.g. EGA, VGA etc.)p0

p1Examples:âlColor☺:=☺UTSTANDARD()☺►☺&&☺returns standard color integerp0
p1âlColor := UTSTANDARD()+128►☺&& returns standard color integerp0
p1âÜ╪╓ß·&& and sets the blink attributep0
p1âVIPRNT(10,10,"Waiting...!",lColor)p0

p(Returns:âInteger☺representing☺the☺CLIPPER color number.p0

```
K⌠x1MÖ╚VERTEX☺CLIPPER☺LIBRARY☺0.9☺USER'S☺GUIDE☺☺PAGE☺167☺☺FPx0☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺

╜☺K1M182Px0☺☺☺☺☺☺☺☺
UNSELECTED()

p,Function:âUNSELECTED()☺-->☺integerp0☺☺☺☺☺☺☺

p=Synopsis:âReturns☺a☺color☺integer☺for☺UNSELECTED settingp0

p1âused by functions ATT() and PRNT()p0

p(Returns:ânumeric☺color☺integerp0☺☺☺☺☺☺☺☺☺☺
```

```
VERTEX CLIPPER LIBRARY 0.9 USER'S GUIDE  PAGE 168  p0☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺

╜☺K1M182p0☺☺☺☺☺☺☺☺☺
IMPORTANT VIDEO FUNCTION INFORMATION¶☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺

p1I have put together a host of videofunctions which are supposed to help in wri
p1applications which use the peculiarities of the Video Cards ( VGA, XVGA etc. )
p1can buy nowadays and give You the chance of using higher resolutions and thusp
p1more info on the screen.p0


p1The provided Videofunctions are all written in Assembler language and provide
p1access to Video Card interrupts.p0

p1The functions will make use of documented features of the video cards and shou
p1therefor work for all video cards that are able to support the screen modes th
p1functions will try to switch to.p0

p1The general videofunctions will use the INT10h BIOS call to switch modes. Foll
p1modes are being used:p0

p100h - 13hp0

p1The documentation to these modes You may find in the Microsoft Press series of
p1books. Before You use any of these general videofunctions make sure that thep0
p1installed videocard can make use of the requested modes. Using the general fun
p1whilst having a videocard installed that does not support the requested modes
p1however result in the worst case in a System-Lockup ( generally the functions
p1don't work ).p0

p1Be aware though that Your Video Card MUST support the display mode You try top
p1select. Usually the functions just don't do anything however in some extreme r
p1cases You may hang the System.p0

p1So use the functions wisely.p0
```

```
╜☺K1M182P☺0☺☺☺☺☺☺☺☺
■GK╤A☺T0)WN¶↕☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺

p,Function:âVIATT(☺<expN1>☺☺.☺.☺<expN5>,☺☺[<expC1>] ) -->p0

p1Synopsis:âChange☺attribute☺☺of☺area☺☺of☺the☺screenp0

p1â<expN1>..<expN4>ßscreen coordinates to change attributes ( top,p0
âp1left, bottom,p0
âp1Ü╪╒ßright )p0

p1â<expN5>  new attribute to set area top0

p1â<expC1>  optional character to fill area withp0

p1âThe functions writes the string direct to VIDEO MEMORY and isp0
âp1thus VERY FAST!!! However it doesn't work in all modes (somep0
âp1graphics modes ). It works with all video modes using eitherp0
âp1B000h or B800h as base address ( e.g. EGA, VGA etc. )p0

p1Examples:âVIATT(☺10,☺10,☺20,☺60,☺16,☺"╪"☺☺)p0☺☺☺☺☺☺☺☺☺

p1Returns☺☺:ânothing.p0☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺
```

K1M182P

VICLS()

p1Function:âVICLS([<expN>], [<expC>]) --> NILp0

p1Synopsis:âClears the current screen with optional color and characterp0

p1â[<expN>]╪an optional color-number which is used to clear thep0
âp1current screenp0

p1â[<expC>]╪an optional character which is used.p0

p1âThis is a low level function to clear the screen and does notp0
âp1depend upon the CLIPPER color scheme. The numeric color codep0
âp1however is the same that CLIPPER supports sofar.p0

p1Examples:âVICLS()╪╔ß·&& clear screen with current colorp0
p1âVICLS(48)╔ß·►☺&& clear screen black on cyanp0
p1âVICLS(48, CHR(176))·►☺&& clear screen black on cyan withp0
p1âÜ╪╔ß·&& character "░"p0

p1Returns:âNothingp0

VICOLGRID()

p1Function:âVICOLGRID(<expN1>, <expN2>)p0

p1Synopsis:âDraws colored blocks in SETCOLORS() functionp0

p1â<expN1> top row coordinatep0

p1â<expN2> top left coordinatep0

p1âThe function will draw a number of blocks on the screen inp0
âp1SETCOLOR() attributes. Used by MFSETCOLOR() functionp0

p1Returns:âNothingp0

Function: VTFADE(<expN1> .. <expN4>, <expC1>, <expN5>) --> NIL

Synopsis: Fades a screen variable off the screen

<expN1> .. <expN4> screen coordinates

<expC1> screen variable to fade off the screen

<expN5> time delay to use (0 - 20)

Returns: Nothing

VIGETMODE()

Function: VIGETMODE() --> array

Synopsis: Returns Video System Info

p1âThe function will post Video System information into an array 5p0
âp1elements long with following information:p0

p1âarray[1]╪hex string of current screen modep0

p1âarray[2]╪available columns (integer)p0

p1âarray[3]╪available rows (integer)p0

p1âarray[4]╪current page (integer)p0

p1âarray[5]╪0 for MONO, 1 for COLOR adapterp0

Examples: Array := VIGETMODE()

Returns: array if called successfully, otherwise NILp0

```
K⌠☺1M☺Ö☺╚VERTEX☺CLIPPER☺LI☺IBRARY☺0.9☺USER'S☺GUIDE☺☺PAGE☺174☺☺FP☺0☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺

╜☺K☺1M☺182P☺0☺☺☺☺☺☺☺☺
#L┤K╤☺1┤$☺CGA☺♦)☺K☺H¶↕☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺

p,Function:âV☺ISCGA()☺☺-->☺logicalp0☺☺☺☺☺☺☺☺☺☺☺☺☺☺

p-Synopsis:â☺Checks☺if☺active☺Display☺is☺in☺CGA☺mode.p0


p(Returns:â☺.T.☺active☺Display☺Adapter☺is☺in CGA mode;p0
p1âotherwise .F.p0
```

```
K1M182Px0
VV_ISEGA()

p1Function: VV_ISEGA() --> logical p0

p1Synopsis: Checks if active Display is in EGA mode.p0


p1Returns: .T. active Display Adapter is in EGA mode;p0
p1otherwise .F.p0
```

VIISHERC()

Function: VIISHERC() --> logical

Synopsis: Checks if active Display is in HERCULES mode.

Returns: .T. active Display Adapter is in HERCULES mode
otherwise .F.

```
K1M182Px0

V_ISMCGA()

p,Function:âV_ISMCGA() --> logicalp0

p1Synopsis:âChecks if active Display is in MCGA mode.p0


p1Returns:â.T. active Display Adapter is in MCGA mode;p0
p1âotherwise .F.p0
```

```
VERTEX☺CLIPPER☺LIBRARY☺0.9☺USER'S☺GUIDE☺☺PAGE☺178☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺

╜☺K1M182Px0☺☺☺☺☺☺☺☺
VIISMDA()☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺

p,Function:âVIISMDA()☺☺-->☺logicalp0☺☺☺☺☺☺☺☺☺☺☺☺☺☺

p=Synopsis:âChecks☺if☺active☺Display☺is☺in☺MDA☺modep0


p(Returns:â.T.☺active☺Display☺Adapter☺is☺in MDA mode;p0
p1âotherwise .F.p0
```

```
K1M182Px0
VVISVGA()

p,Function: VVISVGA()  --> logicalp0

p=Synopsis: Checks if active Display is in VGA modep0


p(Returns: .T. active Display Adapter is in VGA mode;p0
p1âotherwise .F.p0
```

K1M182Px0

VIPRNT()

p1Function:âVIPRNT(<expN1>, <expN2>, <expC1>, <expN3>) --> NILp0

p1Synopsis:âWrite string to row/col with attributep0

p1â<expN1> - rowp0
p1â<expN2> - colp0
p1â<expC1> - String to writep0
p1â<expN3> - color attribute of stringp0

p1âThe functions writes the string direct to VIDEO MEMORY and isp0
âp1thus VERY FAST!!! However it doesn't work in all modes (somep0
âp1graphics modes). It works with all video modes using eitherp0
âp1B000h or B800h as base address (e.g. EGA, VGA etc.)p0

p1Examples:âViPrnt(10,10,"Hello World",23)p0

p1Notes:âDoes not change cursor location or color setting.p0

p1Returns:ânothing.p0

```
╜☺K1M182Px0☺☺☺☺☺☺☺☺
VISETMODE()

p1Function:âVISETMODE( <expC1> ) --> logicalp0

p1Synopsis:âSets a video card to specified Modep0

p1â<expC1>  numeric value ( in HEXADECIMAL notation ) representingp0
âp1Mode to setp0
âp1Ü╬Video card top0

p1âThis function is an all around video set function. It can set ap0
âp1videocard to ANY mode provided it supports it. If You use thisp0
âp1function, first of all make sure that Your MONITOR supports thep0
âp1new resolution as some are so high in their display resolutionp0
âp1that they may damage Your monitor if they are not supported.p0
âp1The function will accept the mode in decimal and convert it top0
âp1hexdecimal.p0

p1âPLEASE REFER TO YOUR MONITOR AND VIDEO CARD MANUAL FORp0
âp1FURTHER INFORMATION IF BOTH SUPPORT THE TYPE OF DISPLAYp0
âp1MODE YOU DESIRE TO SELECT!!!p0

p1âWatch out for the resolutions only supported by MULTISYNC orp0
âp1MULTIFREQUENCY monitors ( they are generally the high HEXp0
âp1numbers > 20h )p0

p1âHowever take care in selecting a mode that works fine with Yourp0
âp1system.p0

p1Examples:âVISETMODE( "7" )ß·&& sets VGA videocard to mode 07hp0

p1âVISETMODE( "29" )ß·&& sets VGA card with ET4000p0
p1âÜ╬╔ß·&& into 800x600 graphics modep0

p1âVISETMODE( "53" )ß·&& sets VGA card with TVGA 8900p0
p1âÜ╬╔ß·&& into 132x25 textmodep0


p1Returns:â.T. if called successfully, otherwise .F.p0
```

≠[K╤SLIDEOFF()H¶↕☺☺☺

K1Function:âVISLIDEOFF(☺)☺☻☻>☺NILp0☺☺☺☺☺☺☺☺☺☺☺☺☺

K1Synopsis:âSlides☺a☺screen☺variable☺off☺the☺screenp0

p1â<expN1> .. <expN4> Screen coordinates to slide offp0

p1â<expC1> variable containing the saved screenp0

p1â<expN5> speed of slide..........>p0

p1âThe function will slide a saved screen area off the screen. Use ap0
âp1function that saves the screen area first. Then use the slider....p0

K1Returns:âNothingp0☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺